John Theodore MacDermott

PROJECT CONTROL USING THE PROBLEM-ORIENTED LANGUAGE PROJECT

PROJECT CONTROL USING THE PROBLEM-ORIENTED LANGUAGE PROJECT

by

LTJG JOHN THEODORE MACDERMOTT, CEC, USN

B.S., United States Naval Academy
(1965)

submitted in partial fulfillment

of the requirements for the

degree of

Master of Science in Civil Engineering

at the

Massachusetts Institute of Technology

August 1967

ABSTRACT

PROJECT CONTROL USING THE PROBLEM-ORIENTED
LANGUAGE PROJECT

by

LTJG JOHN THEODORE MACDERMOTT, CEC, USN

Submitted to the Department of Civil Engineering on August 21, 1967 in partial fulfillment of the requirements for the degree of Master of Science in Civil Engineering.

The problem-oriented language PROJECT, of which this research forms a part, was developed in order to facilitate the use of CPM and PERT on any project which can be described in terms of work items. The area with which this thesis deals is that of monitoring a project's progress. As is true of the rest of the PROJECT subsystem, no special computer knowledge is necessary to use this feature since all instructions to the computer are couched in the language of project management.

The preparation of PROJECT's progress control capability was marked by two phases. The first phase centered around the development of the measures to be used as monitors. The concept underlying this developmental work was that of comparison. Actual work progress and costs are compared with estimated work progress and costs in order to determine a project's status. In addition to being able to determine the status of the project as a single entity, the status of any subset of activities--called sub-networks-- can be determined also.

The second phase of this research was concerned with interfacing the monitoring feature with PROJECT. It was during the implementation phase that the algorithms developed earlier were translated into workable computer programs, compatible with PROJECT.

Thesis Advisor: Professor Albert G. H. Dietz

Title: Professor of Civil Engineering

ACKNOWLEDGEMENTS

There have been many direct and indirect participants in the preparation of this paper. For their assistance, I wish to thank

The Civil Engineer Corps of the United States Navy for sponsoring my fifteen month tour of duty at M.I.T.;

Professor Albert G. H. Dietz for his sincere interest and help in the formulation of the progress control monitor and for his help in the preparation of this thesis;

Professor William H. Linder for suggesting the area of investigation, for his constructive criticism of my work and for his help in finding many elusive programming errors;

Mr. Robert L. Daniels for his helpful attitude which never flagged even in the face of my most ridiculous errors and for his wit which made this undertaking a pleasure;

Mr. Marcus H. Ray for his insight, his willingness to help and for his most valuable contribution which made possible the graphical display;

Mr. Bernard-Andre Genest for his preparation of The Use of ICES PROJECT;

Messrs. Robert Feldman and John O'Doherty of the Massachusetts Bay Transportation Authority for willingly giving up their free time and for making available the MBTA Computer Facility;

Professor Daniel Roos, Director of M.I.T.'s Civil Engineering Systems Laboratory, for making this facility available, and,

The Massachusetts Bay Transportation Authority for their sponsorship of PROJECT.

John T. MacDermott
August 21, 1967
Cambridge,
Massachusetts

TABLE OF CONTENTS

CHAPTER ONE

Introduction

Introduction

PROJECT is a problem-oriented computer language which was designed and developed by William H. Linder in the Department of Civil Engineering at the Massachusetts Institute of Technology. PROJECT is a subsystem of the Integrated Civil Engineering System (ICES). Briefly described, the capabilities of ICES PROJECT are:

1. to represent a project in terms of work items and time;
2. to make projections (over a project's lifetime) of incurred costs and resource consumption;
3. to adjust project schedules to satisfy time or resource constraints; and
4. to provide a method of monitoring project progress.

The last item in the list of PROJECT capabilities--project control-- is to be the subject of this paper. The concern with project control stems from the fact that a prime function of an organization which undertakes a project of any sort is the management and control of the resources associated with that project. Increased emphasis on rapid completion, brought about by keen marketing competition and high interest rates on borrowed money, and the technology "explosion" which has permeated nearly every field of endeavor are but two of several significant factors which have helped to increase both the difficulty and importance of managing and controlling capital investment projects.

In his book Control and Management of Capital Projects, J. W. Hackney suggests that the techniques for the "control of projects naturally group themselves into those related to:

.Capital cost -- estimating and controlling the money invested

•Time -- planning, scheduling and monitoring for smooth progress

and toward early completion

•Value -- pre-determining and controlling income as related to investment, operating costs and risks"[1]

In PROJECT, the progress control monitor is based on work progress and financial progress.

Because it was recognized that the project manager is already burdened with a good deal of information, a prime consideration in the development of the project control monitor was that it provide worthwhile information yet be simple. The goal, then, was to provide the project manager with a single value which is an indicator of the project's status. The philosophy behind PROJECT's progress control feature is that of "management by exception." This means that actual work and cost progress are compared with estimated work and cost progress to indicate those areas of the project which may prevent the project as a whole from finishing (1) on schedule and (2) within its budget.

The Road Ahead

The next chapter of this paper deals with the mathematical formulation of the status index and its components. Chapter Three discusses the philosophy of computing the status index -- the assumptions and limitations inherent in it. In the fourth chapter, the commands for using the project control feature are reviewed. Chapter Five is devoted to a sample problem which illustrates how all phases of this work tie together as a usable part of the PROJECT subsystem. The last section, Chapter Six, suggests some areas in which more work might be done and concludes by

---

[1]Control and Management of Capital Projects, J. W. Hackney, John Wiley & Sons, Inc., 1965, pg. 3.

presenting some general observations. In Appendix A, the interested reader will find listings of the programs pertinent to PROJECT's progress control feature. Appendix B briefly describes the PROJECT command structure.

CHAPTER TWO

The Status Index and Its Components

2.1 The Work Progress Component
2.2 The Financial Progress Component
2.3 The Status Index

The Status Index and Its Components

"Can we finish on time and remain within the project's budget?" This question, often raised by project managers, points to two items which are critical to any project--time and money. Concern with work and financial progress has provided the motivation for the development of a status index based on these items. The status index, formed from the product of factors representing work and financial progress components, forms the basis of PROJECT's progress control feature.

2.1 The Work Progress Component

Before developing the work progress component of the status index, it is necessary to present some definitions.

1. Original project duration: T

   This is the original estimate of the project duration--the result of the first unrevised, but complete, planning schedule. Except for a situation to be discussed later, T is a project constant.

2. Revised project duration: T'

   This is the latest revised estimate of project duration. It is based on data reported from the project site and is equal to the time from the project start to the present plus the most recent estimate of how long it will take (again, measuring from the present) to finish the project.

3. Elapsed time: t

   t is the time from project start to the present.

4. Remaining time: t'

   t' represents the latest revised estimate (measuring from the present) of how much longer it will take to finish the project. As the project advances, t' is estimated over successively smaller time intervals and should, therefore, be increasingly accurate. At the project start, T = T' = t'.

5. Initial amount of time recoverable: K

   K is the amount of time which can be "made up" over the project

lifetime by accomplishing the appropriate activities as quickly as possible. K is estimated at the start of the project; it is a parameter evaluated for each project and is the amount of time recoverable when t = 0.

6. Current amount of time recoverable: J

This is the amount of time which can be made up at any point in the project's lifetime. J is a function of the original project duration, the total time recoverable and the elapsed time -- J = f (T,K,t).

As the project advances, less and less time can be made up by crashing activities. Just how the amount of time which can be made up behaves is not clear, but to a first approximation it is reasonable to assume that it decreases linearly. With this assumption of linearity, let

$$J = K(1 - \frac{t}{T}) \qquad (2.1)$$

When J is less than (T' - T) the project will take more time to complete than can be made available even if all succeeding activities are crashed.

Figure 2.1 shows graphically what has been presented thus far. The significance of the parameter K becomes clear at this point. As the value which fixes the time recoverable curve (here, the straight line represented by J), K must be carefully determined and ought not be modified unless the project schedule is changed. Figure 2.1 suggests a number of other noteworthy points. First, for any value of time t (where t is less than T) the value (ordinate) of the corresponding point on the line J represents the time which could be made up by making a maximum effort. Next, the value of the ordinate at a is the amount of time that the project is behind schedule. Finally, if the point a and the line J happened to be coincident, the project can be finished on time if a maximum effort is made.

In terms of the previous definitions, the work progress component of

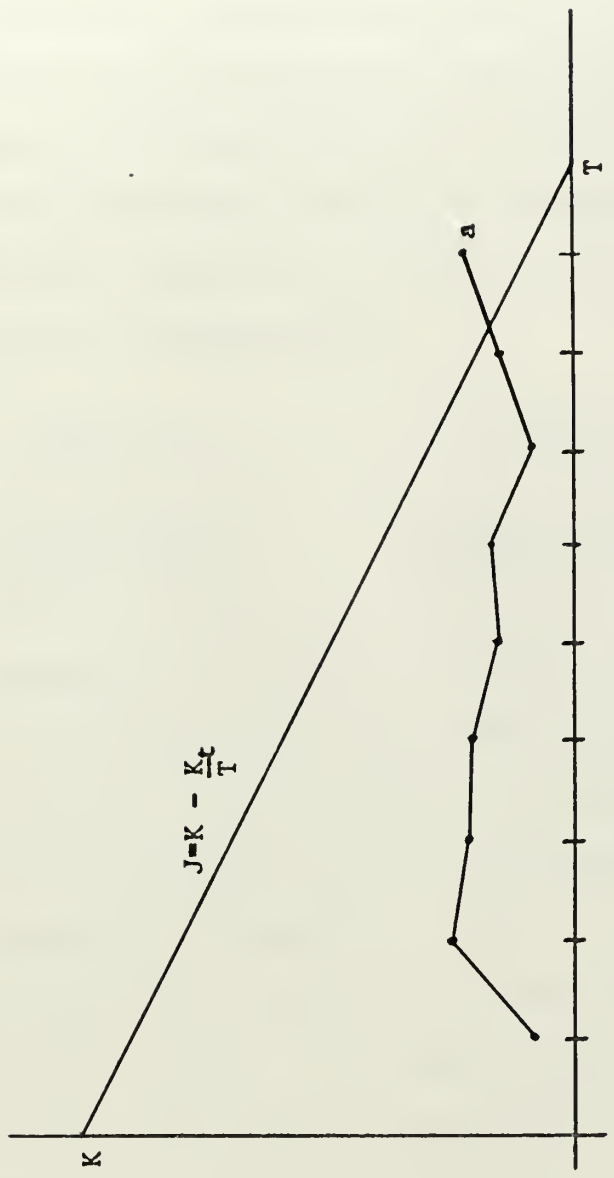


Figure 2.1: The time recoverable curve

the status index is

$$I_W = \frac{J - (T' - T)}{J} \qquad (2.2)$$

When actual work progress agrees exactly with what was scheduled, $I_w = 1$. If the actual work is further along than was anticipated when the schedule was prepared, I_w reflects this fact by assuming a value greater than 1. Should actual work represent less of an accomplishment than was planned for a particular time, I_w will be less than unity. These conditions are summarized in the following table.

Value of I_w	Implication
less than one	behind schedule
equal to one	exactly on schedule
greater than one	ahead of schedule

In the definition of the project parameters T and K, circumstances were alluded to under which these values might change. If it happens that J is less than (T' - T) it is unwise to continue to deal in terms of the current network because, based on the estimate of time recoverable, there is no possibility of finishing by time T. In this situation, the network must be modified to reflect (1) the work actually finished and (2) changes in the work plan. The project network should be modified to represent this "new" project and revised values of the parameters T and K should be associated with the revised project.

When t = T, J = 0 and the work progress measure 2.2 goes to infinity. This means that the work progress component (and hence the status index) cannot be computed for the last day of the project. This does not constitute a major limitation since the value of the status index when the project is essentially complete is not of great interest. It is of in-

terest to note the effect of the difference (T' - T) on the behavior of the work progress component. When the project is on schedule, I_W is one, suggesting that T' = T. If the project is behind schedule, (T' - T) is greater than zero; the quantity (T' - T) acts to decrease the numerator, making it smaller than the denominator. In this case, should the numerator become negative (that is, J less than (T' - T)), the situation is infeasible with respect to time and the project must be reconsidered. Should the project be ahead of schedule (T' - T) is negative; this quantity now acts to increase the value of the numerator in 2.2. In this case, I_W will grow larger than one, behaving as advertised. There may be some concern with the fact that the denominator of 2.2 decreases while the numerator remains relatively constant (approximately equal to (T' -T)). This concern is unwarranted since (T' - T) strictly positive means that the project will finish on day T' which occurs before day T. Mathematically speaking, this means that J is bounded from below by a value larger than zero, hence the denominator does not go to zero.

2.2 The Financial Progress Component

As was indicated at the outset, the general concept behind the status index and its components is to compare actual progress with extimated progress. In the financial component of the status index, this goal is achieved by relating, for a specific time t during the project's lifetime, (1) the amount of money actually spent, (2) the amount scheduled to have been spent and (3) the original estimate of the total project cost.

Before proceeding with the discussion, it is necessary to consider the following terms:

1. Let C represent the original estimate of the total project cost.

2. Let A be the amount of money scheduled to be spent by time t.

3. Let A' represent the amount of money actually spent at time t.

For the sake of clarity, a graphical interpretation of these symbols appears on the following page in figure 2.2. Two points can be made concerning this diagram. First, the actual spending curve is known only for the completed part of the project; therefore, for time subsequent to t this curve is represented by a dotted line to indicate that it is an "educated guess." Of course, since t approaches T, the "educated guess" is based on an ever decreasing time interval and should, therefore, be increasingly accurate. The second point to be made is that the terminal points of the actual spending curve and the estimated project cost curve need not be coincident--that is, the project may cost more (or less) than the original estimate.

In consonance with the "definitions" of the symbols A, A' and C, the financial progress component is

$$I_c = \frac{C - (A' - A)}{C} \qquad (2.3)$$

The cost progress measure (2.3) behaves in precisely the same way as the work progress measure (2.2). A value of I_c equal to one implies that scheduled expenditures and actual expenditures agree exactly--that is, $A' = A$. A value of I_c larger than one means that $(A' - A)<0$ and less money has been spent by time t than was originally estimated. Similarly, if I_c is less than unity, then at time t more money has been spent than was planned and $(A' - A)>0$. The following table summarizes the implications of various values of the financial progress component of the status index.

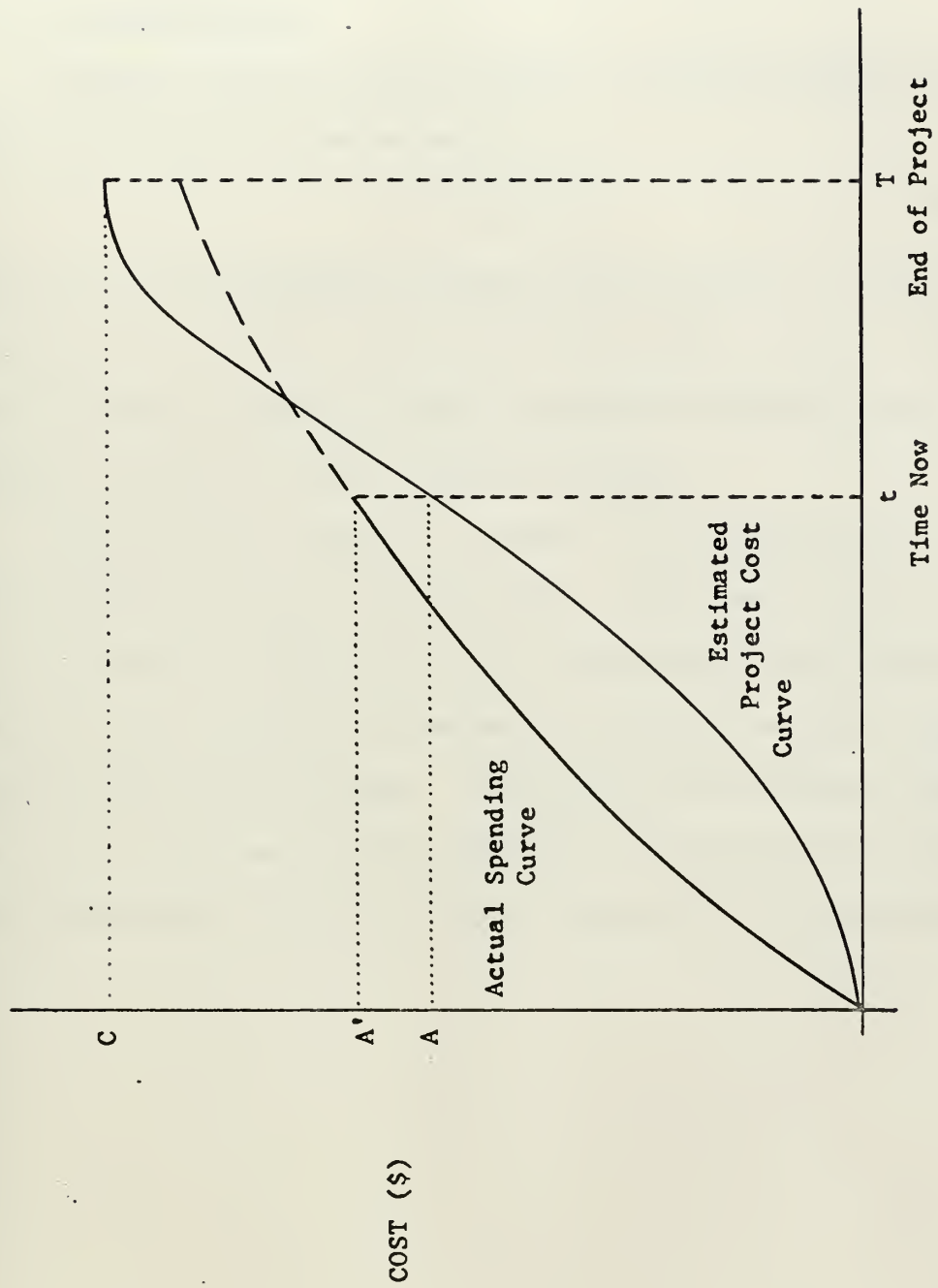


Figure 2.2: Actual and estimated project spending curves

Value of I_c	Implication
less than one	overrunning budget
equal to one	exactly on budget
greater than one	underrunning budget

2.3 The Status Index

PROJECT's progress control feature, the status index, is the product of the cost and time measures:

$$SI = I_c \times I_w \qquad (2.4)$$

The status index indicates agreement between planned and actual progress when its value is one. A value larger than unity indicates actual progress is ahead of planned; a value less than one suggests that actual progress has fallen behind scheduled progress.

Since the status index is a product, it is necessary to be aware of the cancelling effects of its formation. For example, if $I_c = 0.5$ and $I_w = 2.0$ then $SI = (0.5) \times (2.0) = 1.0$. This value would imply that the project was on schedule with respect to both cost and work progress--certainly this is not correct. The problem just described constitutes one aspect of the more general question of interfacing the status index with ICES PROJECT. This is to be the subject of the next chapter.

CHAPTER THREE

The Status Index and ICES PROJECT

3.1 Data Structure
3.2 Computation of the Cost Component
3.3 Computation of the Work Progress Component
3.4 Project Sub-networks

The Status Index and ICES PROJECT

Providing PROJECT with a progress monitor can be thought of as a two step operation. The first step, considered in the last chapter, involved developing a suitable monitor. The second step centers around the problem of interfacing the monitor with PROJECT. As will be shown, more is involved in the implementation of the status index than just programming considerations. The philosophy behind the computations, together with the assumptions and limitations inherent in the monitoring feature, are the subject of this chapter.

3.1 Data Structure

Each project may have as many as eight disk files associated with it. These files, called NAME1 through NAME8 (viz. WALTHAM3) are used to store data on the particular paoject under consideration.

PROJECT's progress monitoring feature deals with disk file NAME8. The structure of this file consists of three levels: two levels are used for bookkeeping and can be thought of a pointers, the third level is used for data storage. The first time reported progress data is stored for a particular project, NAME8 is established by creating a first-level logical record ten words* long. The first word of this record contains the address of a second-level logical record which is NOACT words long, where NOACT is the number of activities in the network. Each word of the second logical record contains a zero when it is first created. When progress data is reported for a particular activity, a third-level record corresponding to that activity is created; this record is also ten words long. The address of this third level record is

---

* A"word" in computer terminology is a unit of storage.

stored in the second level logical record in the position corresponding to the activity under consideration.* The logical record on the third level contains the reported progress data. The first word of this record contains the cost (if reported), the second word contains the reported activity finish date (if reported) and the third word contains the start date (if reported). All dates are stored as project workdays. If any of the possible data items--cost, finish date or start date--are omitted, the corresponding word of the activity data record is not changed; this means that until a cost, finish or start is reported, the corresponding word in the activity data record contains a zero.

In order to add some measure of clarity to the foregoing description of NAME8, the following example and diagrams are presented. Consider a project having seven activities. Assume progress is first reported on the third activity: this creates disk file NAME8 for the project and stores the data as shown in figure 3.1 on the following page. Arrows are used to denote the pointer arrangement alluded to earlier. If progress is subsequently reported on the second activity, NAME8 would be structured as shown in figure 3.2.

It seems that much space is unused in NAME8. This is the case; however, it is anticipated that the uncommitted space will be used in connection with resource allocation and leveling in the near future.

3.2 Computation of the Cost Component

As discussed earlier, the cost component of the status index is

---

* The activities comprising a network are ordered topologically. Thus, if a network consists of activities 5, 10, 20, 25 and 30, activity 5 is the first activity, activity 10 is the second, and so on. If progress is reported on the third activity, the third word of the second-level logical record will contain the address of a third-level logical record which itself contains the reported progress data for, in this example, activity 20.

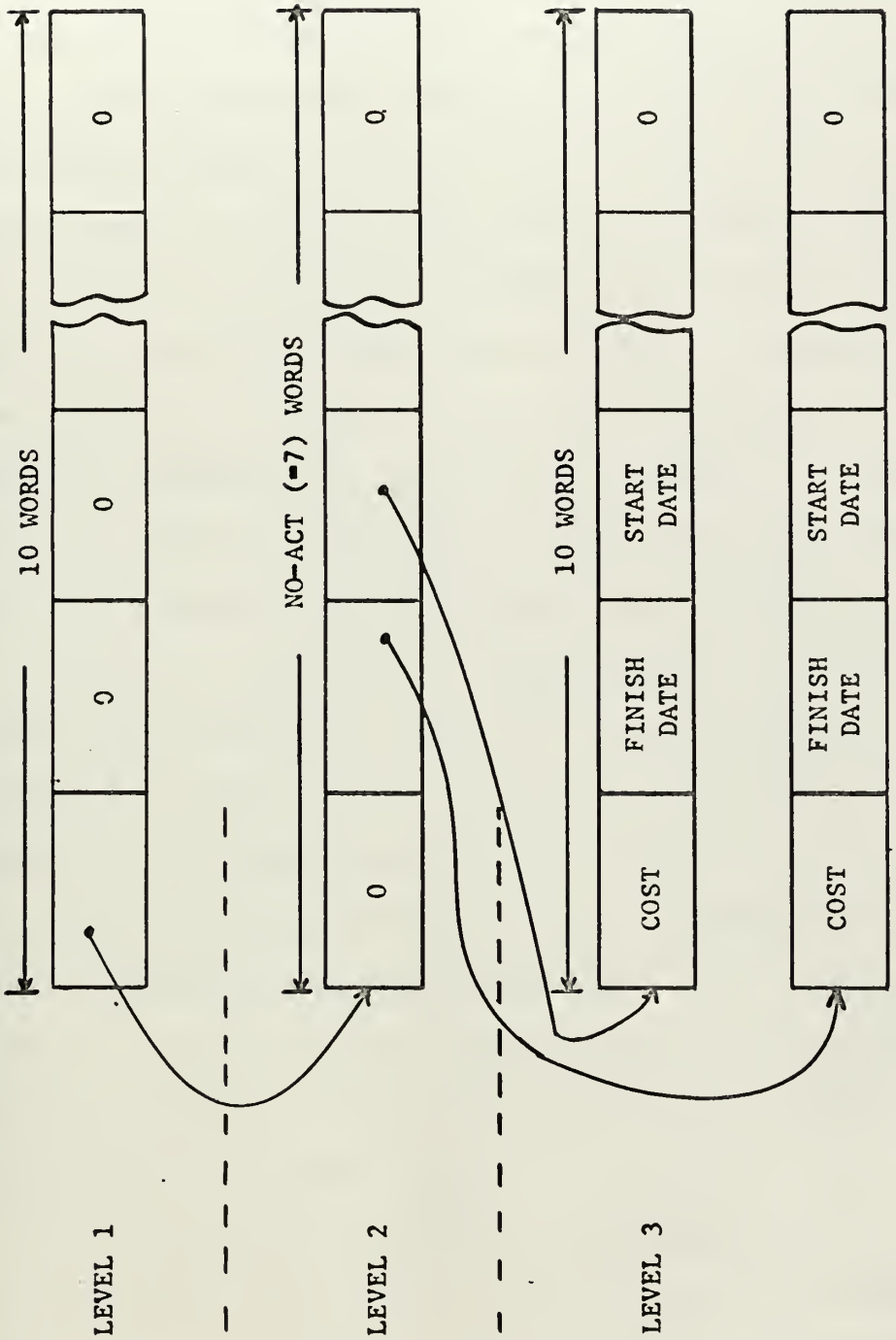


Figure 3.2: Name 8 after progress is reported on two activities

$$I_c = \frac{C - (A' - A)}{C}$$

In order to carry out this computation for time t, it is necessary to know (1) the sum of the estimated costs for each activity (this is C), (2) the amount of money planned to be spent by time t (this is A) and (3) how much money was actually spent by time t (this is A'). There are two very significant points to be made. First, costs are based strictly on activity costs; no effort is made to take into account overhead costs. The second point is that if an activity is in progress at the time a computation is called for a linear portion of the cost assigned to that activity is taken. For example, assume that a particular activity has been assigned an estimated cost of $600; if a computation is requested when the activity ought to be two-thirds complete, this activity's contribution to the estimated cost to date figure will be $400.

The method of determining the estimated cost to date for the project is most clearly explained by using a flow chart of operations, as found in figure 3.3 on the next page. The actual cost to date is computed in exactly the same fashion except that if an actual cost for an activity has not been reported, it is assumed that the estimated cost for that activity is accurate and is a reasonable figure on which to base the computation. If it is found that there is neither an actual nor an estimated cost for a particular activity this activity makes no contribution at all to the actual cost to date figure.

It is clear that a limitation of the system is that an estimated cost must be available on at least one activity if the cost component or the status index is to be computed. This requirement obtains because if no estimated costs are available, the cost component would be $\frac{0}{0}$, hence

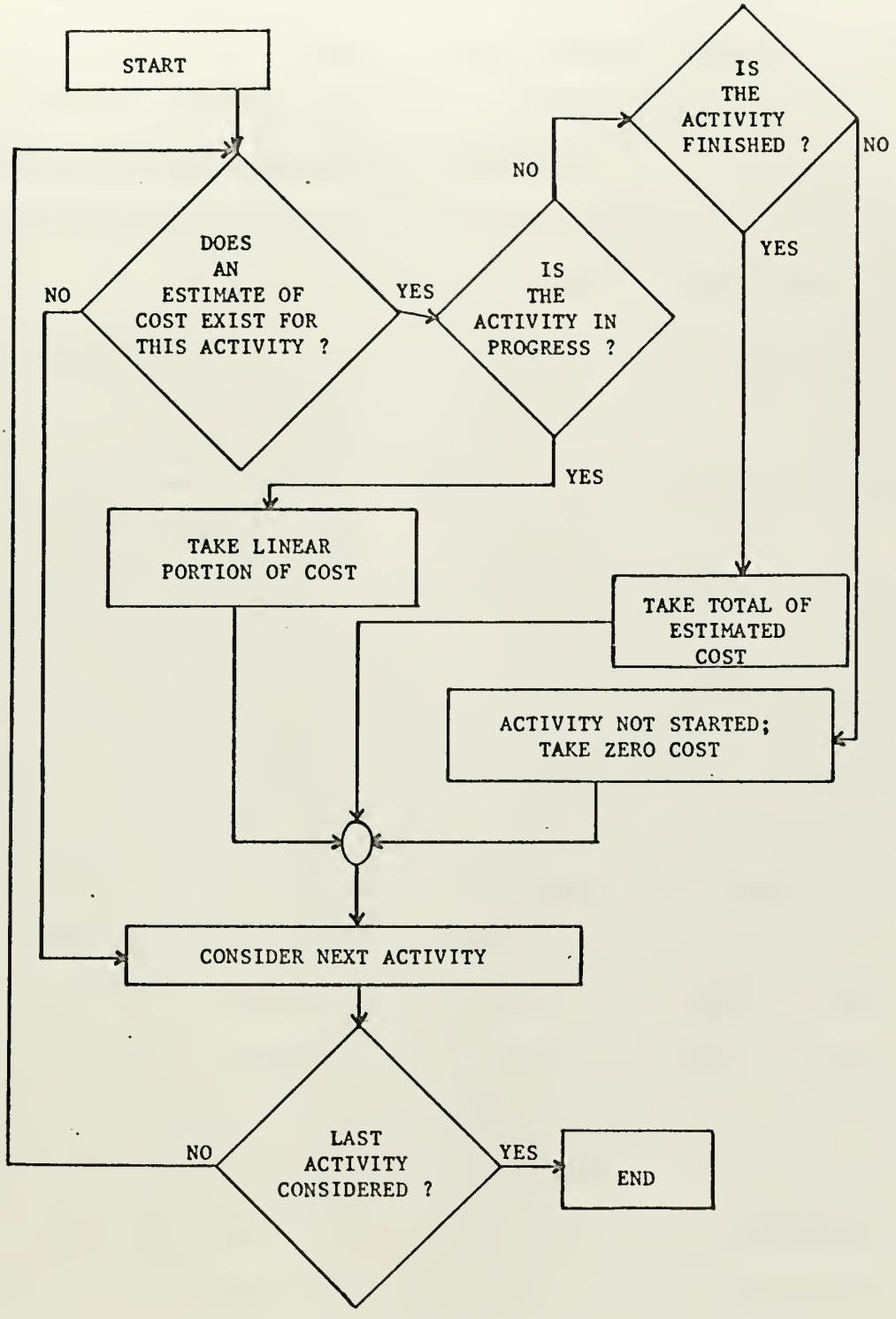


Figure 3.3: Determining the estimated total cost to date

indeterminant. A second limitation is that if actual costs are not reported for activities, then in equation 2.3 $A = A'$ and $I_c = 1.0$. Thus, if actual costs are not reported, the status index is nothing more than the work progress component tagged with a different name.

3.3 Computation of the Work Progress Component

The work progress component of the status index is

$$I_w = \frac{J - (T' - T)}{J}, \text{ where } J = K(1 - \frac{t}{T}).$$

The determination of J is straight-forward since K, t and T are known. When $t = T$, as was pointed out earlier, J is zero and the computation of I_w is not allowed; in PROJECT, a check is always made to insure that J has a non-zero value.

Assuming that J is valid, it remains only to determine the value of T', the latest revised estimate of the job duration. This computation is based on the reported progress information for the network activities which has been stored in NAME8. Reported progress data on an activity may consist of an estimated total cost, a start date, a finish date or any non-repetitive combination of the these items. Since costs play no role in computing the work progress component, there are four cases to be considered for each activity in the network: (1) Neither a start nor a finish has been reported. In this case, it is assumed that the duration as originally input is accurate and that the activity will be executed at the time specified by the working schedule. (2) A start date has been reported but a finish date has not. In this situation, the original activity duration is used to compute a finish date for the activity; the activity under consideration is then fixed in time by these two dates. (3) A finish date has been reported but a start date has

not. This case is analogous to the preceding one in that the original activity duration and the reported finish date are used to compute a start date; these dates then serve to fix the activity in time. (4) Both a start and finish date have been reported. Here, the given dates fix the activity in time and are used to compute a new activity duration.

The primary function of the reported start and/or finish data is to fix certain activities in the network with respect to time. This "fixed schedule" information is used as input data to perform the standard forward pass algorithm; this forward flowing of the network yields the latest revised estimate of the job duration, T'. Note that a backward pass is not made using the fixed schedule information. This is because the backward pass is useful for determining (1) the network's critical path and (2) total and free float associated with each activity; neither of these items is of particular concern when determining the work progress component of the status index.

3.4 Project Sub-networks

Thus far all discussion has centered around performing progress monitoring computations for the project network as a whole. It is entirely reasonable, quite likely, in fact, that the project manager may be deeply concerned about the performance of one or several subsets of network activities. PROJECT provides several methods of identifying such subsets--referred to here as project sub-networks. Moreover, PROJECT's progress control feature is capable of performing status index computations for sub-networks in a fashion similar to the method used for the network as a whole. The word "similar" is stressed in the preceding sentence because the network and sub-network computations differ on one

basic point as will be explained in the following paragraph.

In performing a status index computation for a sub-network, all activities not a part of the subset being considered are assigned a zero duration. The idea here is to retain activity interdependencies while eliminating information which may be prejudicial to the computation for the sub-network. A consequence of assigning zero durations to activities not in the sub-network is that reported progress data on activities which are a part of the sub-network cannot justifiably be used to fix these activities in time as was done when the computation was made for the network as a single entity. The following example will serve to clarify this point. Consider the network shown in figure 3.4a. Suppose the sub-network of interest consists of activities 10 and 40. The sub-network can be pictured as in figure 3.4b. The sub-network duration is four time units. Now assume (1) that activity 40 started on schedule (day 8) but finished one day late (day 10) and (2) that sub-network computations for determining T' were based on the same algorithm used for the network as a whole. Under these conditions, activity 40 would be constrained to start on day 8 and finish on day 10; this would yield a sub-network duration of nine time units when, in fact, it took but five time units to complete the activities of interest. In order to avoid this misleading situation, the reported progress data for a sub-network of activities is used only to determine revised durations for activities comprising the subset. Thus, unless both a start and finish date are given for an activity, it is assumed that the original duration is accurate and may be used as a basis for computing the latest revised estimate of the sub-network duration. A very significant ramification of this basis for computing the sub-network T' is that so long as the

durations of activities in the sub-network do not change, I_w for that subset of activities will be unity, regardless of whether the activity was completed between its scheduled dates. For clarification of this point, consider the network shown in figure 3.4c. Suppose activity 20 actually took five time units to complete instead of four as planned. Both the latest revised estimate and the original estimate of the sub-network (activities 10 and 40) duration would be five time units and are not affected by the delay in activity 20 since this activity is not a part of the sub-network of interest.

In addition to the method of determining the sub-network duration, the method of arriving at the amount of time recoverable for a sub-network must be discussed. This computation is quite straight-forward. The original estimate of the sub-network duration is divided by the original estimate of the project duration; this quotient is multiplied by the estimate of time recoverable for the project and the result is taken as the time recoverable for the sub-network. For example, say project has a duration of forty days and that a sub-network of interest has a duration of ten days. If the estimate of time recoverable for the project were eight days, then the time recoverable for the sub-network would be $(\frac{10}{40}) \times 8$ or two days.

In closing this discussion of sub-networks and the computations related to them, it should be pointed out that cost computations--both estimated and actual--are carried out in exactly the same way as they are performed for the network as a whole.

This chapter has dealt with the question of interfacing the project monitor with ICES PROJECT. Pertinent algorithms, limitations and assump-

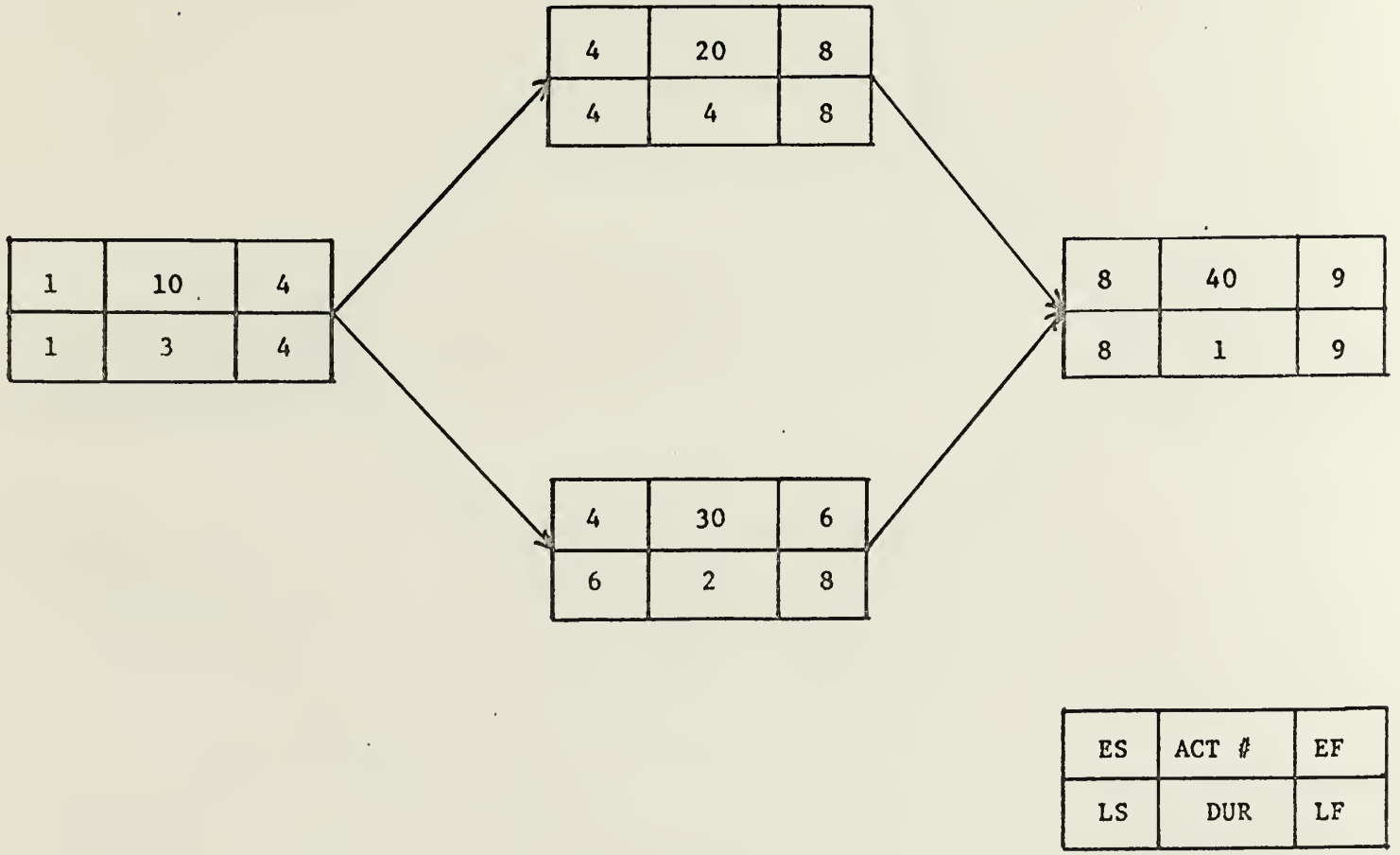


Figure 3.4a: A sample network

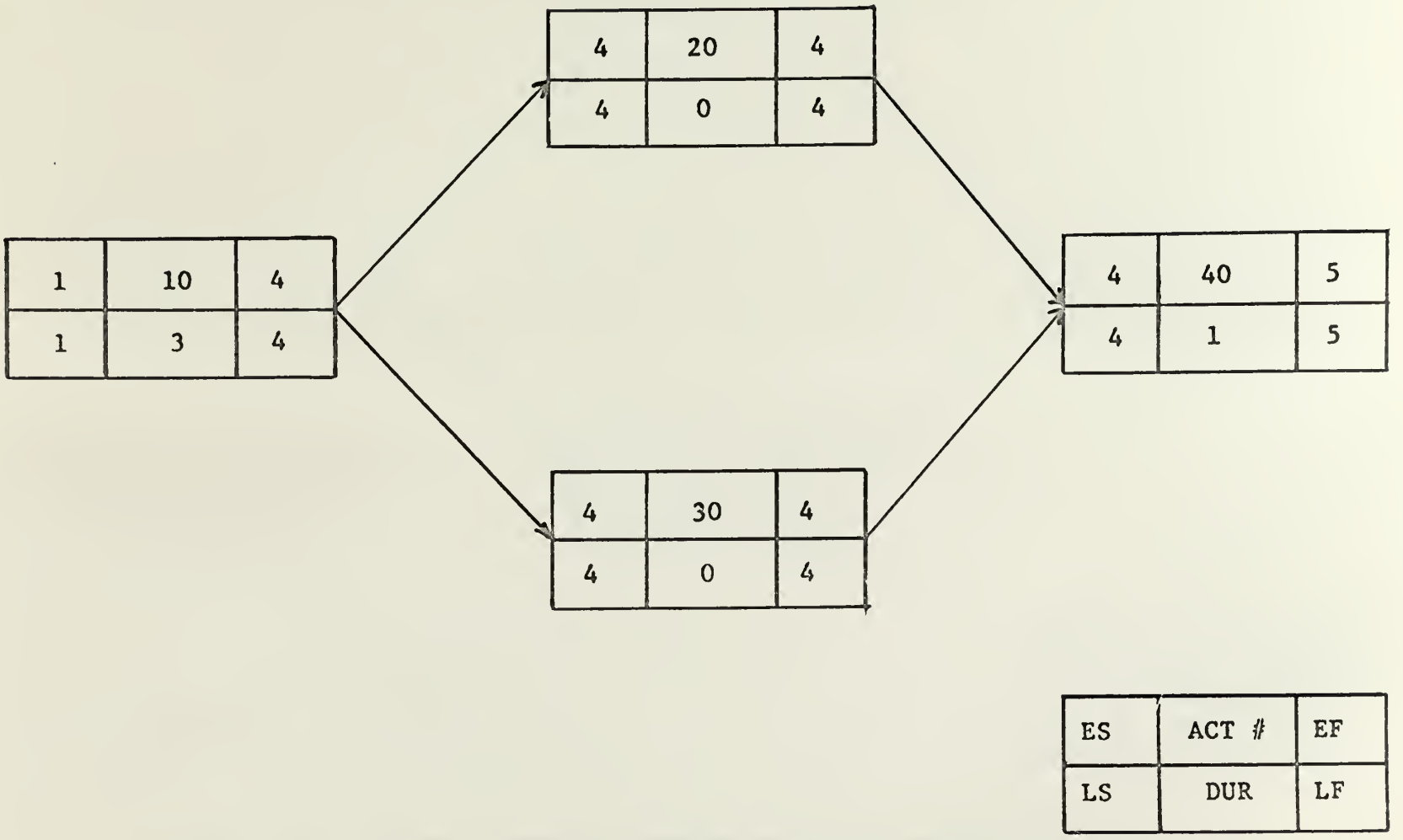


Figure 3.4b: The sub-network consisting of activities 10 and 40

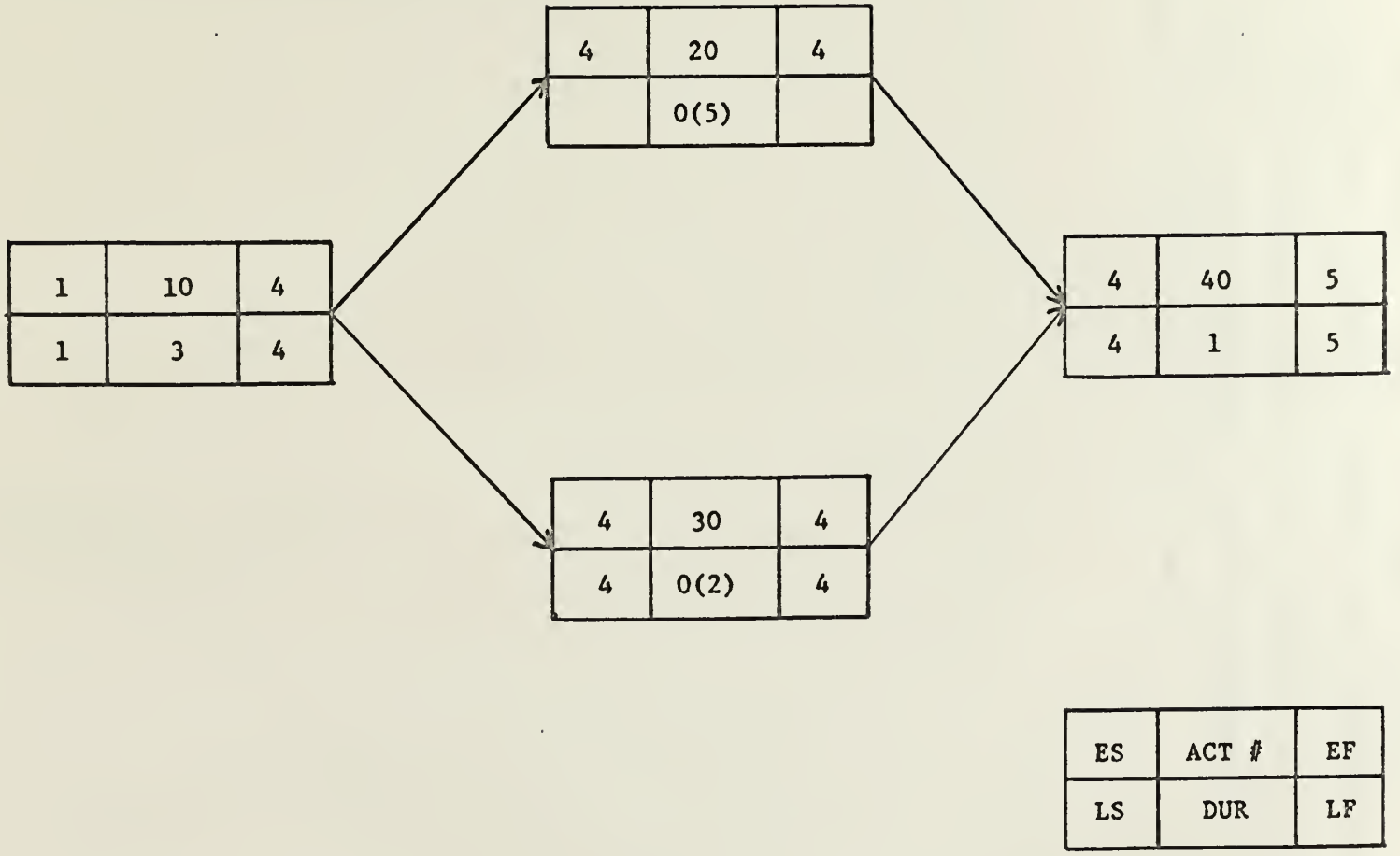


Figure 3.4c: Effect of duration changes for activities not in the sub-network

tions have been discussed in detail. The computer programs, illustrating how these algorithms, limitations and assumptions have been used in this development, may be found in Appendix A of this paper. The next section, Chapter Four, presents the commands associated with PROJECT's progress control feature.

CHAPTER FOUR

The Commands Associated with PROJECT's Progress Monitoring Feature

The Commands Associated with PROJECT's Progress Monitoring Feature

A basic part of any problem-oriented computer language is the commands which define the user-subsystem interface. For this reason, it is of value to give a brief description of how command requests are handled by ICES PROJECT. The following steps effectively summarize this procedure:

Step 1:	User prepares and submits command
Step 2:	The ICES Command Interpreter "reads" the command
Step 3:	On the basis of what was read by the Command Interpreter, certain programs are executed
Step 4:	Icetran programs compute requested results and output them to the user.

Illustrated and discussed in this chapter are the commands which have been written for use with PROJECT's progress control feature. The purpose is to examine the capabilities of the commands so that the interested user may, if necessary, modify them to suit his own needs.

4.1 The PROJECT Command Structure

Before embarking on a discussion of the specific commands which constitute a part of the progress monitoring feature, it is worthwhile to make some general observations concerning the format of command composition in ICES PROJECT. All PROJECT commands are composed of three parts:

1) an operation name,
2) a data lable and
3) an object phrase

The operation name is the first word of any command; it indicates the general type of operation to be carried out. The data lable specifies whether the command applies to all project networks handled by PROJECT at a given time or to a particular project network. Finally, the object phrase, which may consist of one or more words and numbers, serves to specify more precisely the nature of the work to be done by a command. For a definitive discussion of the commands associated with the subsystem, the reader is referred to The Use of ICES PROJECT[2]; the brief overview presented here should, however, provide the background necessary for the ensuing examination of the commands specifically related to the progress monitoring feature.

The basic operation names used by PROJECT's monitoring feature are: ASSIGN, PRINT, PLOT and REPORT. In the remaining portion of this chapter, the basic operation names are examined with a view towards building useful commands for project control.

4.2 Commands--INPUT

It was indicated in Chapter Two that two classes of data must be provided in order to use PROJECT's progress monitor. The first type of information is the reported activity progress data. The command to begin storage of this data is Command 1:

---

[2]The Use of ICES PROJECT, Bernard-Andre Genest (Editor), Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Mass., 1967.

REPORT ('projname') { PROGRESS / COST / START / FINISH } (AS OF) date*

The cards which follow Command 1 are the activity progress data cards. The first number on each card must be the activity number (if an activity-on-node network) or numbers (if an activity-on-arrow network). As indicated above, Command 1 may take any of four forms depending upon which reporting option is desired. Designate option 1a to be PROGRESS, 1b to be COST, 1c to be START and 1d to be FINISH. Consider command option 1a:

REPORT ('projname') PROGRESS (AS OF) date

On the data cards, following the activity number may appear (in any order) the word START followed by the start date, FINISH followed by a finish date and COST followed by an amount. The data may be input without using the modifiers START, FINISH and COST by preparing the data cards so that the order of information is activity number(s), start date, finish date and cost. When the "no modifier" input option is used, both a start and finish date must be specified; moreover, these dates must be calendar dates (e.g. 1 July 1967). With the "no modifier" input form, the cost data for a particular activity is optional and may be omitted. For the command option 1 with modifiers any non-repetitive combination of the words COST, START and FINISH is acceptable. Thus, the PROGRESS command may be used to store information on only start dates, only finish dates, finish dates and costs, etc.

Progress reporting options 1b, 1c, and 1d (corresponding to COST, START, and FINISH, respectively) constitute subsets of the PROGRESS

---

* See Appendix B

command in that they are used when data is reported on just costs, only start dates or just finish dates. For these options, too, the identifier (the word COST, START or FINISH) may be omitted.

The last card in a series of data cards should have the word LAST punched on it. This insures that the ICES command interpreter will pass on to the subsequent command in the normal fashion.

A significant point should be made regarding the reporting of progress. There is no reason why estimates of activity finishes, starts or costs cannot be submitted. Assuming that such data is prepared with care, this practice will result in a more comprehensive comparison between actual and planned work than would result if this procedure is not followed.

The second class of input data referred to earlier consists of a single number. It is the amount of time recoverable and was denoted by the symbol K in Chapter Two. Once the amount of time recoverable has been estimated, Command 2 is used to store it:

ASSIGN ('projname') TIME (RECOVERABLE) K $\left\{\begin{matrix}\text{DAYS}\\\text{WEEKS}\end{matrix}\right\}$,

where k represents the time recoverable in units of days or weeks, with days assumed if neither time unit is specified. Should the user neglect to assign it, fifteen percent of the job duration is automatically taken as the amount of time recoverable; a message indicating this is printed out. If, after the initial assignment of time recoverable, it is desirable to modify this estimate, Command 2 should be employed again.

4.3 Commands--Output

PROJECT's progress control feature provides several very useful output features. All of them stem from two basic operation names: PRINT and PLOT.

If the value of the status index or any of its components is desired for a single date, the appropriate option of Command 3 ought to be used:

PRINT ('projname') INDEX (FOR) { STATUS / TIME / COST } (AS OF) date ({ phase / select / list })

It was pointed out earlier that if the status index is requested, both of its components will be examined to see if they differ from unity by more than one-tenth; if there is an offending component, its value is printed. As is indicated in Command 3 and as was discussed in Chapter 3, it is possible to request the status index or its components for a pre-defined phase or on the basis of some user-defined selection option. For more detail on these selection options, the reader is referred to Chapter 10 of The Use of ICES PROJECT.

Should the value of any project monitor be desired as it varies between two specified dates, Command 4 would be used:

PRINT ('projname') INDEX (FOR) { STATUS / TIME / COST } BETWEEN date 1 and date 2 ({ phase / select / list })

If the interval between the first date and the second is less than 100 workdays, increments are in steps of five workdays; if the interval is larger than 100 workdays, increments are in steps of 10 workdays.

Of great value in examining both project and sub-network progress trends is a graphical display of the status index (or the requested component) versus time. Command 5 was developed just for this purpose:

PLOT ('projname') INDEX (FOR) { STATUS / TIME / COST } BETWEEN date 1 and date 2 ({ phase / select / list })

The comments regarding the incremental stepping apply here as they did

for Command 4. Moreover, in both Commands 4 and 5 it makes no difference whether the starting date is date 1 and the finish date is date 2 or vice versa. A significant point regarding Commands 4 and 5 is that they permit the computation of a status index for any point in time which falls within the project's lifetime. Thus, the project manager may request a computation for some future time in order to get a projection of project status if present progress trends continue. Whenever progress is reported using any of the options associated with Command 1, the date which is a part of the REPORT command is stored. Any computation which is requested for a date later than the date accompanying the most recent REPORT command is considered to be a projected value. Commands 3 and 4 make no distinction between projected and actual computations; the PLOT command, however, prints actual values with a "*" and projected values with a "-". A final item of interest concerning the PLOT command is that it has been written with a linear fill-in feature. The effect of this feature is to fill in the space between successive ordinates on a strictly linear basis. The linear fill-in does not distort the display in any fashion but serves to make trends indicated by the graph easier to spot.

This chapter has provided some background on the structure of commands in PROJECT and has discussed the commands currently available for use with PROJECT's progress control capability. The next section, Chapter 5, presents a sample problem which illustrates how the progress control capability may be used in practice.

CHAPTER FIVE

A Sample Problem: The Southwest Plant

A Sample Problem: The Southwest Plant

All comments made thus far on PROJECT's progress control feature have been of either a developmental or theoretical character. The purpose of this chapter is to demonstrate how the progress control feature can be used on an actual project.

5.1 The Project

The task is to build a sewage disposal plant on a small island in Massachusetts Bay. Sewage will be piped from Boston to the disposal plant through an underwater pipeline. On the island, the sewage will pass through a mechanical pulverizer where the solids will be reduced to small particles; then the waste will be treated chemically prior to its being released into the sea.

5.2 Transportation

The contractor must provide transportation for all of his men, materials and equipment to the island. The City of Boston has provided docking facilities on the mainland for the contractor's use.

5.3 Major Items of Construction

The SW PLANT project has the following major components:

1. An underwater pipeline must be laid from the island station to the Boston shore.

2. Chemical treatment and pulverizing equipment must be ordered and installed.

3. A steel frame superstructure, in which the pulverizer and chemical treatment units will be mounted, must be erected.

4. A permanent boat landing and breakwater must be built as a part of the project.

5. An electric generator must be ordered and installed on the island.

6. A small station house must be built to house the resident operating personnel, the system controls and the generator.

5.4 The Network

The SW PLANT project will be described by an activity-on-node network and will consist of the activities and events listed in figure 5.1. Note that in addition to activity descriptions and durations, extimated activity costs are also provided for each activity. Figure 5.2 shows the SW PLANT network; note the special arrow relationships. The project schedule is given in figure 5.3.

5.5 Monitoring SW PLANT PROGRESS

The project manager has determined that the SW PLANT project could be completed in 32 work days instead of the alloted 38 if a maximum effort were made. Thus, the time recoverable for this project is six days; this information is stored using the command

ASSIGN 'SW PLANT' TIME RECOVERABLE 6 DAYS

as indicated in figure 5.4.

Owing to a special concern with activities related to piping and those activities concerned with the procurement and installation of the generator and chemical treatment equipment, the project manager has, at the outset, defined two phases as indicated in figure 5.4.

Progress in first reported on activities in the sample network on 11 January 1967. Since work started on 2 January, seven working days have elapsed. The data following the REPORT command in figure 5.4 illustrates three ways in which progress can validly be reported. The data for activity 210 shows the "no modifier" option; activity's 310 data illustrates how dates relative to the project start can be used in the command; finally, the data for activity 410 illustrates that when

```
        STORE  'SW PLANT'  NETWORK
$
$       EVENTS
$  NODE NO.          DESCRIPTION
      100       'START PROJECT'
      200       'SPECIAL ITEMS PROCURED'
      300       'TRANSPORTATION ARRANGED'
      400       'ISLAND LANDING COMPLETED'
      500       'CONSTRUCTION COMPLETED'
      600       'TESTING BEGINS'
      700       'UNDERWATER PIPELINE LAID'
     1000       'PLANT COMPLETED'
$
$       ACTIVITIES
$                                                          ACTIVITY    ESTIMATED
$  NODE NO.          DESCRIPTION                          DURATION    DOLLAR COST
      110       'GET PIPE-LAYING EQUIP. & LAY PIPE'          27      COST   22000
      210       'TRANSPORT GENERATOR'                         1      COST    1000
      220       'TRANSPORT CHEM. TREATMENT EQUIPMENT'         2      COST    1000
      250       'INSTALL CHEM. TREATMENT EQUIP.'              2      COST    4000
      310       'CLEAR LANDING & CONST. WHARF'                2      COST   16000
      330       'PUT IN CRANES'                               1      COST    2000
      340       'PUT IN BREAKWATER'                          20      COST   80000
      410       'CLEAR BLDG. SITE & PLACE FNDS.'             10      COST   25000
      420       'TRANSPORT CONSTRUCTION EQUIPMENT'            1      COST    5000
      450       'PUT UP STATION HOUSE'                        1      COST    2000
      460       'ERECT SUPERSTRUCTURE'                       20      COST   10000
      470       'ROUGH IN PIPING'                             4      COST    5000
      510       'MAKE FINAL PIPING CONNECTIONS'               1      COST    2000
      520       'INSTALL VALVES AND GAUGES'                   1      COST    1000
      530       'MAKE FINAL ELECTRICAL CONN.'                 1      COST    1000
      540       'PUT IN WIRING'                               3      COST   10000
      610       'TEST ELECTRICAL SYSTEM'                      1      COST    1000
      620       'TEST PULVERIZER'                             1      COST    1000
      630       'TEST CHEMICAL EQUIPMENT'                     2      COST    2000
      710       'TIE PIPELINE TO ISLAND STATION'              1      COST    1000
```

Figure 5.1

PROJECT "S.W. PLANT"

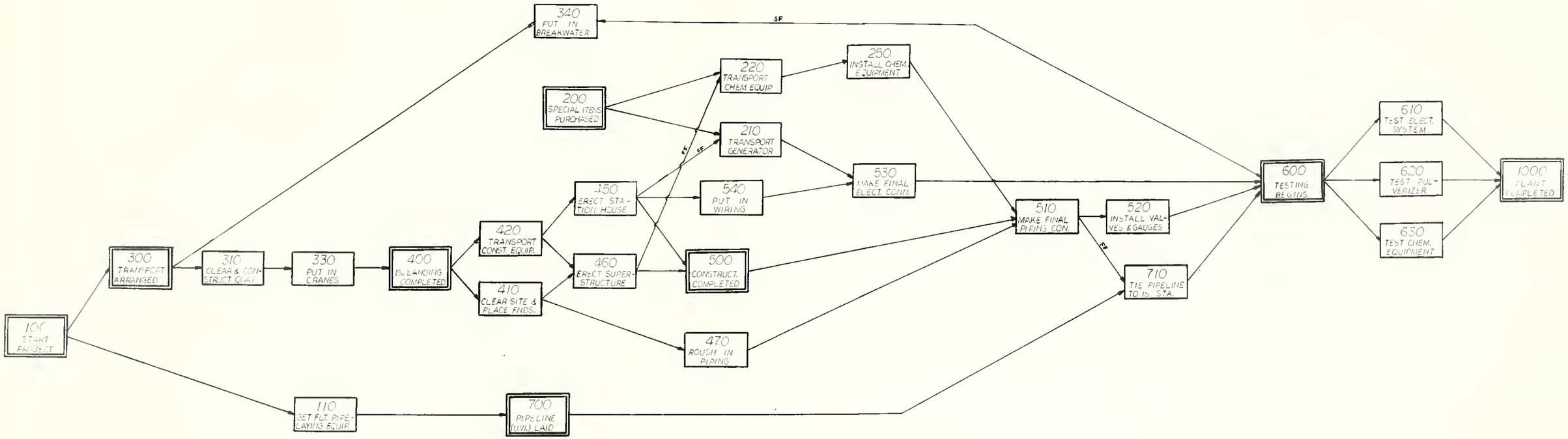


Figure 5.2

```
ASSIGN 'SW PLANT' START 1 JANUARY 1967

         PROJECT SW PLANT HAS BEEN ASSIGNED TO START  ON  DAY    1 JAN 1967

PRINT 'SW PLANT' SCHEDULE

                    ************************************
                    * SCHEDULE FOR PROJECT 'SW PLANT' *
                    ************************************

          PROJECT DURATION IS    38 WORK DAYS,  WORK WEEK IS 5 DAYS
          WORK IS SCHEDULED TO START ON  2 JAN 1967 AND TO BE COMPLETED ON 22 FEB 1967.

                THE PROJECT 'SW PLANT' NETWORK HAS
                        20 ACTIVITIES
                         8 MILESTONE EVENTS

               NO HOLIDAYS FOUND IN SW PLANT HOLIDAY TABLE.

                            EVENT SCHEDULE
                            **************

      EVENT  DESCRIPTION                          EARLY TIME        LATE TIME

C       100  START PROJECT                        2 JAN 1967        2 JAN 1967
                                                        1                 1
        200  SPECIAL ITEMS PROCURED                2 JAN 1967       14 FEB 1967
                                                        1                32
C       300  TRANSPORTATION ARRANGED              2 JAN 1967        2 JAN 1967
                                                        1                 1
C       400  ISLAND LANDING COMPLETED             5 JAN 1967        5 JAN 1967
                                                        4                 4
        500  CONSTRUCTION COMPLETED               16 FEB 1967       20 FEB 1967
                                                       34                36
C       600  TESTING BEGINS                       21 FEB 1967       21 FEB 1967
                                                       37                37
        700  UNDERWATER PIPELINE LAID              8 FEB 1967       20 FEB 1967
                                                       28                36
C      1000  PLANT COMPLETED                      22 FEB 1967       22 FEB 1967
                                                       38                38

                          END OF EVENT SCHEDULE
                          *********************
```

Figure 5.3

ACTIVITY SCHEDULE
****

SCHEDULE

START TIMES ARE FIGURED FOR THE MORNING OF THE WORK DAY
FINISH TIMES ARE FIGURED FOR THE EVENING OF THE WORK DAY

ACTIVITIES ARE SORTED ACCORDING TO NODE NUMBERS

'C' IN MARGIN DESIGNATES A CRITICAL ACTIVITY

	ACTIVITY	DESCRIPTION	DURATION	EARLY START	LATE START	EARLY FINISH	LATE FINISH	FREE FLOAT	TOTAL FLOAT
	110	GET PIPE-LAYING EQUIP. & LAY PIPE	27	2JAN67 1	12JAN67 9	7FEB67 27	17FEB67 35	0	8
		PRECEDES ACTIVITY 700							
	210	TRANSPORT GENERATOR	1	6JAN67 5	14FEB67 32	6JAN67 5	14FEB67 32	0	27
		PRECEDES ACTIVITY 530							
C	220	TRANSPORT CHEM. TREATMENT EQUIPMENT	2	14FEB67 32	14FEB67 32	15FEB67 33	15FEB67 33	0	0
		PRECEDES ACTIVITY 250							
C	250	INSTALL CHEM. TREATMENT EQUIP.	2	16FEB67 34	16FEB67 34	17FEB67 35	17FEB67 35	0	0
		PRECEDES ACTIVITY 510							
C	310	CLEAR LANDING & CONST. WHARF	2	2JAN67 1	2JAN67 1	3JAN67 2	3JAN67 2	0	0
		PRECEDES ACTIVITY 330							
C	330	PUT IN CRANES	1	4JAN67 3	4JAN67 3	4JAN67 3	4JAN67 3	0	0
		PRECEDES ACTIVITY 400							
C	340	PUT IN BREAKWATER	20	24JAN67 17	24JAN67 17	20FEB67 36	20FEB67 36	0	0
		ACTIVITY 340 IS A FINAL (SINK) ACTIVITY							
C	410	CLEAR BLDG. SITE & PLACE ENDS.	10	5JAN67 4	5JAN67 4	18JAN67 13	18JAN67 13	0	0
		PRECEDES ACTIVITY 460 470							

Figure 5.3 (Continued)

	420	TRANSPORT CONSTRUCTION EQUIPMENT		1	5JAN67	18JAN67	5JAN67	18JAN67	0	9
					4	13	4	13		
		PRECEDES ACTIVITY	450 460							
	450	PUT UP STATION HOUSE		1	6JAN67	14FEB67	6JAN67	14FEB67	0	27
					5	32	5	32		
		PRECEDES ACTIVITY	210 500	540						
C	460	ERECT SUPERSTRUCTURE		20	19JAN67	19JAN67	15FEB67	15FEB67	0	0
					14	14	33	33		
		PRECEDES ACTIVITY	220 500							
	470	ROUGH IN PIPING		4	19JAN67	14FEB67	24JAN67	17FEB67	18	18
					14	32	17	35		
		PRECEDES ACTIVITY	510							
C	510	MAKE FINAL PIPING CONNECTIONS		1	20FEB67	20FEB67	20FEB67	20FEB67	0	0
					36	36	36	36		
		PRECEDES ACTIVITY	520 710							
C	520	INSTALL VALVES AND GAUGES		1	20FEB67	20FEB67	20FEB67	20FEB67	0	0
					36	36	36	36		
		PRECEDES ACTIVITY	600							
	530	MAKE FINAL ELECTRICAL CONN.		1	12JAN67	20FEB67	12JAN67	20FEB67	27	27
					9	36	9	36		
		PRECEDES ACTIVITY	600							
	540	PUT IN WIRING		3	9JAN67	15FEB67	11JAN67	17FEB67	0	27
					6	33	8	35		
		PRECEDES ACTIVITY	530							
	610	TEST ELECTRICAL SYSTEM		1	21FEB67	22FEB67	21FEB67	22FEB67	1	1
					37	38	37	38		
		PRECEDES ACTIVITY	1000							
	620	TEST PULVERIZER		1	21FEB67	22FEB67	21FEB67	22FEB67	1	1
					37	38	37	38		
		PRECEDES ACTIVITY	1000							
C	630	TEST CHEMICAL EQUIPMENT		2	21FEB67	21FEB67	22FEB67	22FEB67	0	0
					37	37	38	38		
		PRECEDES ACTIVITY	1000							
C	710	TIE PIPELINE TO ISLAND STATION		1	20FEB67	20FEB67	20FEB67	20FEB67	0	0
					36	36	36	36		
		PRECEDES ACTIVITY	500							

* END OF SCHEDULE *
****

Figure 5.3 (Continued)

modifiers are used, the data may come in any order.

In an effort to get a picture of project status in the near future--day 10--the project manager has used another command found in figure 5.4:

PRINT INDEX FOR STATUS AS OF DAY 10

The requested (projected) value of the status index is 0.77; the work progress component is 0.779 and hence, it is also printed. Notice that because it was available from a previous command, the project name could be omitted from the foregoing command.

Also found in figure 5.4 are the commands

PRINT INDEX FOR TIME AS OF DAY 10 and

PRINT INDEX FOR COST AS OF DAY 10.

These commands serve to illustrate that either component of the status index may be requested.

Figure 5.5 is a glimpse into the future of the SW PLANT project. By using the PLOT command and specifying the range of interest to be the first thirty days of the project, the project manager can see at a glance that if present trends continue, he will find himself in an untenable position.

Shifting his attention from the project as a whole, the project manager next requests the status index as of day 10 for the sub-network called PHASE 1; this command is illustrated in figure 5.6 as is the request to

PLOT INDEX FOR STATUS BETWEEN DAY 1 AND DAY 30 PHASE 1

An interesting point can be noted here. Apparently workday 1 does not occur during the execution of the activities in the sub-network PHASE 1. This is indeed the case as the project schedule shows; thus, since the

```
ASSIGN 'SW PLANT' TIME RECOVERABLE 6 DAYS.

THE ESTIMATE OF TIME RECOVERABLE FOR PROJECT SW PLANT IS      6 DAYS.

ASSIGN 'SW PLANT' PHASE 1 210 220 250 LAST

PHASE       1  OF PROJECT 'SW PLANT' HAS BEEN ASSIGNED THE FOLLOWING ACTIVITIES,

                210          TRANSPORT GENERATOR
                220          TRANSPORT CHEM. TREATMENT EQUIPMENT
                250          INSTALL CHEM. TREATMENT EQUIP.


ASSIGN 'SW PLANT' PHASE 2 470 510 520 110 710 LAST

PHASE       2  OF PROJECT 'SW PLANT' HAS BEEN ASSIGNED THE FOLLOWING ACTIVITIES,

                470          ROUGH IN PIPING
                510          MAKE FINAL PIPING CONNECTIONS
                520          INSTALL VALVES AND GAUGES
                110          GET PIPE-LAYING EQUIP. & LAY PIPE
                710          TIE PIPELINE TO ISLAND STATION


REPORT PROGRESS AS OF 11 JAN 1967

ACT 210 6 JAN 1967 9 JAN 1967 2000.
NO MODIFIERS.  ASSUMED INPUT ORDER TO BE START, FINISH, COST (OPTIONAL).

310 START 1  FINISH 3  COST 17000.

ACT 410 FINISH 20 JAN 1967 START 5 JAN 1967 COST 27000.

LAST

PRINT INDEX FOR STATUS AS OF DAY 10

                          **** THE VALUE OF THE STATUS INDEX IS  0.77 FOR DAY   10 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.779 ****

PRINT INDEX FOR TIME AS OF DAY 10
                     **** THE VALUE OF THE WORK PROGRESS COMPONENT IS  0.78 FOR DAY   10 ****

PRINT INDEX FOR COST AS OF DAY 10
                          **** THE VALUE OF THE COST COMPONENT IS  0.99 FOR DAY   10 ****
```

Figure 5.4

PLOT INDEX FOR STATUS BETWEEN DAY 1 AND DAY 30

INDEX DATA WILL BE PRESENTED IN 5 DAY INCREMENTS.

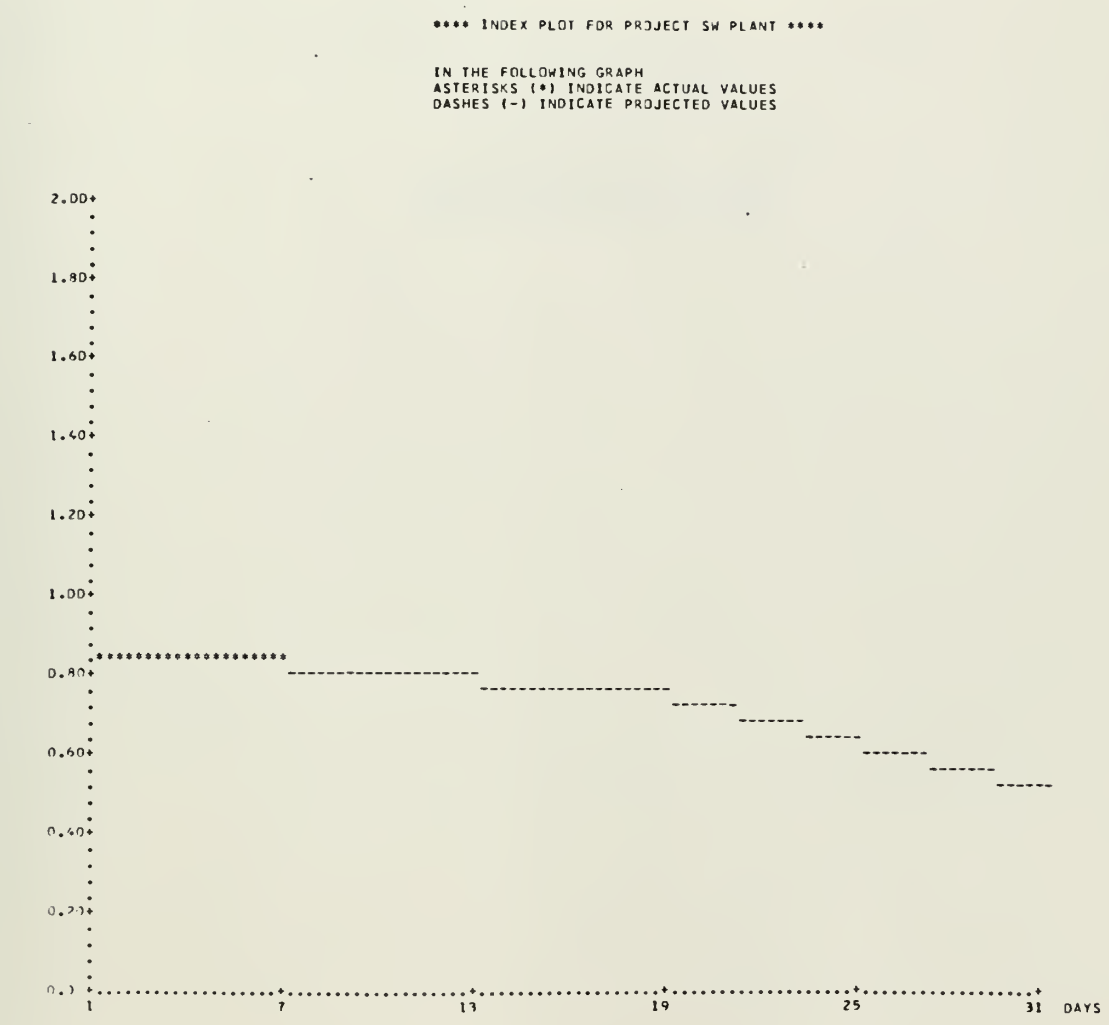


Figure 5.5

```
PRINT INDEX FOR STATUS AS OF DAY 10 PHASE 1

                          **** THE VALUE OF THE STATUS INDEX IS  0.84 FOR DAY   10 ****

        **** NOTE -- VALUE OF COST COMPONENT IS  0.833 ****

PLOT INDEX FOR STATUS BETWEEN DAY 1 AND DAY 30 PHASE 1

INDEX DATA WILL BE PRESENTED IN  5 DAY INCREMENTS.

WORKDAY    1 DOES NOT OCCUR DURING THE EXECUTION OF THIS SUBSET OF ACTIVITIES.  IT IS NOT VALID.


                                    **** INDEX PLOT FOR PROJECT SW PLANT ****

                                    IN THE FOLLOWING GRAPH
                                    ASTERISKS (*) INDICATE ACTUAL VALUES
                                    DASHES (-) INDICATE PROJECTED VALUES
```

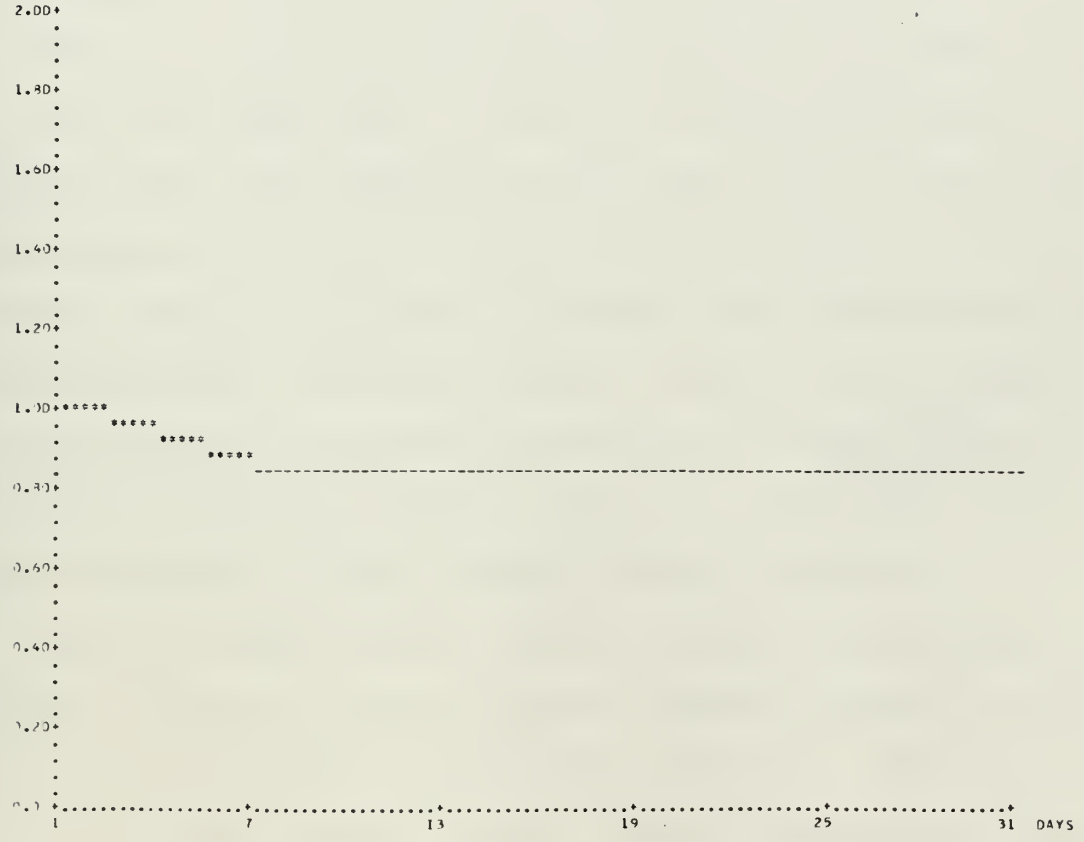


Figure 5.6

sub-network has not started, its status index value is unity by definition. This is indicated on the plot in figure 5.6.

On the twentieth day of the project progress is reported as shown in figure 5.7. One new command is introduced here; it is

PRINT INDEX FOR STATUS BETWEEN DAY 1 AND DAY 20.

Based on incremental steps of five working days, the index is computed and printed in the same fashion as when it is requested for a single date. Notice that the values reported for the status index in figure 5.7 start at day 1 with a lower value than was recorded on the basis of the progress reported on 11 January 1967 (c.f. figure 5.4). The reason for this is that the values recorded in figure 5.7 are based on different progress data than is the index value found in figure 5.4.

Figure 5.8 shows what progress was reported for the project on day 35. Again, the status index is printed for an interval between two specified dates. The projected and actual index values for day 20 which were displayed in figure 5.7 showed the SW PLANT project to be in an unfavorable situation with respect to progress. This information was used to good advantage by the project manager; evidence of this is given by the near-unity status index values--see figure 5.8--which indicate that the project is behaving as planned. The plot of index values over the same period (found in figure 5.9) makes the same indication.

Again, the project manager's attention shifts to the important subnetworks. The plot of the status index for PHASE 2, figure 5.10, shows this subset of activities to be performing quite well. Figure 5.11 is a plot of the status index for PHASE 1; PHASE 1 seems to have a status index which is constant at a value of 0.84. This is a relatively low

```
REPORT PROGRESS AS OF DAY 20

ACT 340 24 JANUARY 1967 21 FEB 1967 8300D.
NO MODIFIERS.  ASSUMED INPUT ORDER TO BE START, FINISH, COST (OPTIONAL).

ACT 460  FIN 33  STA 14  COS 1200D.

ACT 470 COST 350D. STA 15  FIN 18

LAST

PRINT INDEX FOR STATUS AS OF DAY 20

                              **** THE VALUE OF THE STATUS INDEX IS  0.65 FOR DAY   20 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.653 ****

PRINT INDEX FOR STATUS BETWEEN DAY 1 AND DAY 20

INDEX DATA WILL BE PRESENTED IN  5 DAY INCREMENTS.

                              **** THE VALUE OF THE STATUS INDEX IS  0.84 FOR DAY    1 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.834 ****

                              **** THE VALUE OF THE STATUS INDEX IS  0.80 FOR DAY    6 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.807 ****

                              **** THE VALUE OF THE STATUS INDEX IS  0.76 FOR DAY   11 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.770 ****

                              **** THE VALUE OF THE STATUS INDEX IS  0.71 FOR DAY   16 ****

         **** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS  0.717 ****

PRINT INDEX FOR STATUS AS OF DAY 20 PHASE 1

                              **** THE VALUE OF THE STATUS INDEX IS  0.84 FOR DAY   20 ****

         **** NOTE -- VALUE OF COST COMPONENT IS  0.833 ****

PRINT INDEX FOR STATUS AS OF DAY 20 PHASE 2

                              **** THE VALUE OF THE STATUS INDEX IS  1.06 FOR DAY   20 ****
```

Figure 5.7

```
REPORT PROGRESS AS OF DAY 35

220 14 FEBRUARY 1967 14 FEBRUARY 1967
NO MODIFIERS.  ASSUMED INPUT ORDER TO BE START, FINISH, COST (OPTIONAL).

250 FINISH 16 FEB 1967 START 16 FEB 1967

LAST

PRINT INDEX FOR STATUS AS OF DAY 35

                          **** THE VALUE OF THE STATUS INDEX IS  0.97 FOR DAY   35 ****

PRINT INDEX FOR STATUS BETWEEN 1 AND 35

INDEX DATA WILL BE PRESENTED IN  5 DAY INCREMENTS.

                          **** THE VALUE OF THE STATUS INDEX IS  1.01 FOR DAY    1 ****
                          **** THE VALUE OF THE STATUS INDEX IS  1.00 FOR DAY    6 ****
                          **** THE VALUE OF THE STATUS INDEX IS  0.99 FOR DAY   11 ****
                          **** THE VALUE OF THE STATUS INDEX IS  0.99 FOR DAY   16 ****
                          **** THE VALUE OF THE STATUS INDEX IS  0.99 FOR DAY   21 ****
                          **** THE VALUE OF THE STATUS INDEX IS  0.98 FOR DAY   26 ****
                          **** THE VALUE OF THE STATUS INDEX IS  0.98 FOR DAY   31 ****
```

Figure 5.8

PLOT INDEX FOR STATUS BETWEEN 1 AND 35

INDEX DATA WILL BE PRESENTED IN 5 DAY INCREMENTS.

**** INDEX PLOT FOR PROJECT SW PLANT ****

IN THE FOLLOWING GRAPH
ASTERISKS (*) INDICATE ACTUAL VALUES
DASHES (-) INDICATE PROJECTED VALUES

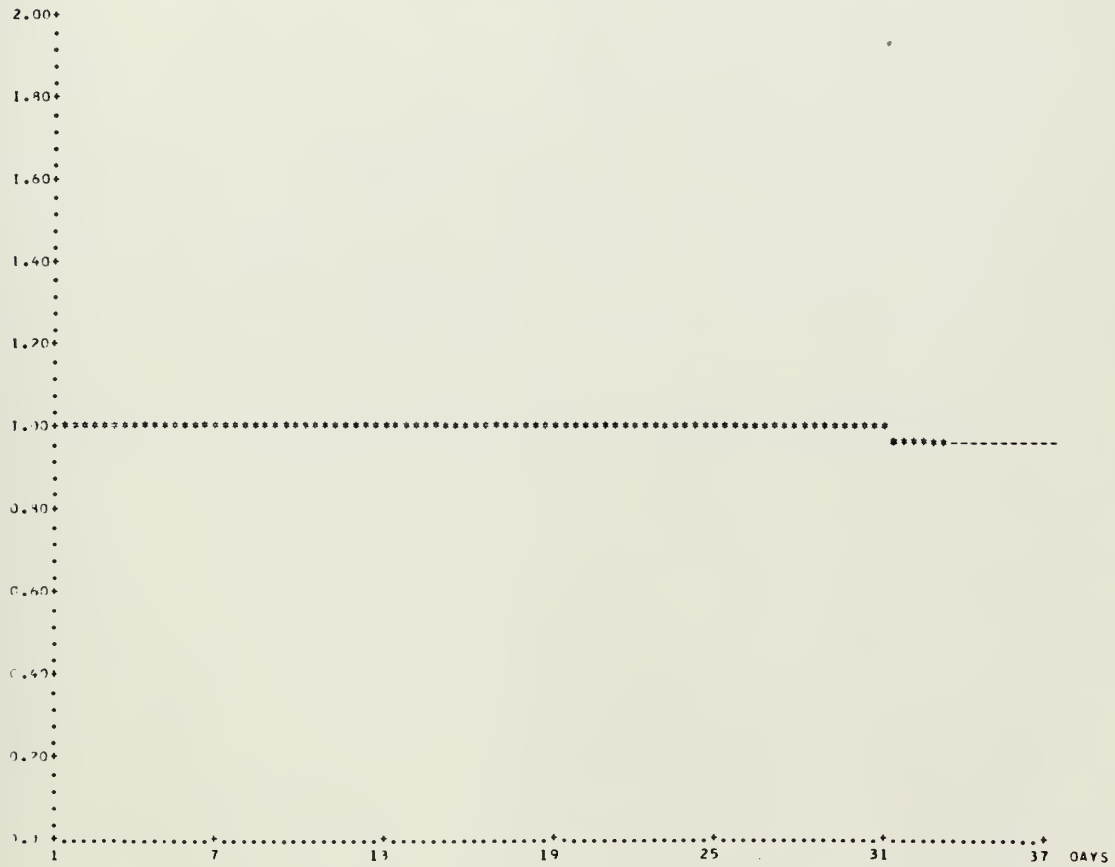


Figure 5.9

```
PRINT INDEX FOR STATUS AS OF DAY 35 PHASE 2

                          **** THE VALUE OF THE STATUS INDEX IS  1.06 FOR DAY   35 ****

PLOT INDEX FOR STATUS BETWEEN 1 AND 35 PHASE 2

INDEX DATA WILL BE PRESENTED IN  5 DAY INCREMENTS.

                                  **** INDEX PLOT FOR PROJECT SW PLANT ****

                                  IN THE FOLLOWING GRAPH
                                  ASTERISKS (*) INDICATE ACTUAL VALUES
                                  DASHES (-) INDICATE PROJECTED VALUES
```

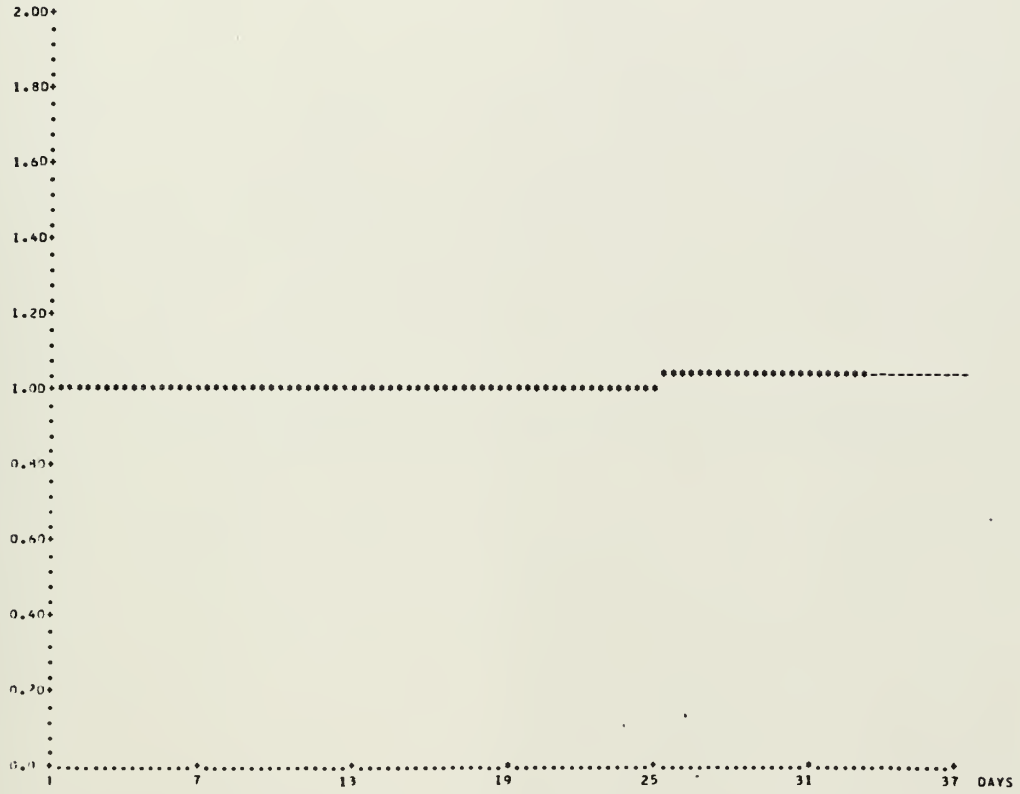
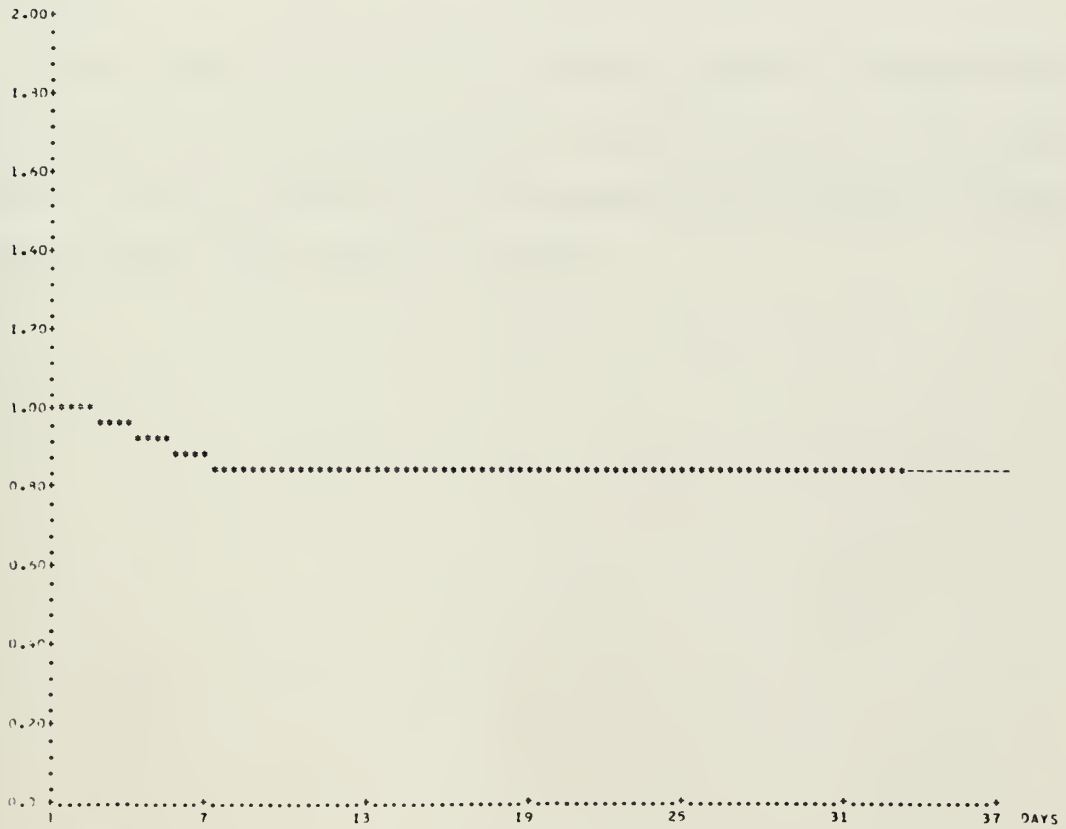


Figure 5.10

```
PLOT INDEX FOR STATUS BETWEEN 1 AND 35 PHASE 1
INDEX DATA WILL BE PRESENTED IN  5 DAY INCREMENTS.
WORKDAY    1 DOES NOT OCCUR DURING THE EXECUTION OF THIS SUBSET OF ACTIVITIES.  IT IS NOT VALID.

                                    **** INDEX PLOT FOR PROJECT SW PLANT ****

                                    IN THE FOLLOWING GRAPH
                                    ASTERISKS (*) INDICATE ACTUAL VALUES
                                    DASHES (-) INDICATE PROJECTED VALUES
```




```
PRINT INDEX FOR STATUS AS OF DAY 35 SELECT ACTIVITIES WITH NODES GT 510

                    **** THE VALUE OF THE STATUS INDEX IS  1.01 FOR DAY   35 ****

FINISH
```

Figure 5.11

value considering that only three days remain in the project's lifetime. It would certainly behoove the project manager to investigate this the activities comprising PHASE 1 very carefully.

The final area of interest to the project manager consists of all network activities with node numbers greater than or equal to 510; this includes activities 510, 520, 530, 540, 610, 620, 630 and 710. The value of 1.01 for the status index (printed below the graph in figure 5.11) indicates that this user-defined subset of activities is on schedule.

5.6 Summary

In this chapter, every working command of PROJECT's progress control feature has been demonstrated on an actual project. It has been shown how the status index and its components may be useful as indicators of problem areas within the project.

CHAPTER SIX

Extensions and Comments

Extensions and Comments

The aim of this chapter is to draw together the several facets of PROJECT's progress control feature. This will be done by first discussing some extensions of the capability and then by making some observations of a general nature.

6.1 Extensions

Calculation of the work progress measure for any sub-network requires that the forward flowing procedure be twice applied: once to determine the planned sub-network duration and the second time to calculate the latest revised estimate of the sub-network duration. Each time the forward flowing takes place, a check is made to ascertain if any special arrow relationships (that is, any arrow relationships which are other than finish-start) are present in the network. Thus, this check is made twice whenever the status index (or the work progress measure) is requested--a time consuming procedure. A worthwhile extension would be to improve the efficiency of the progress monitoring capability by eliminating the redundant check.

In Chapter Two, a situation was described when the existing project schedule ceases to be viable. This can occur when the project will take longer to complete than was originally planned. Under such circumstances, it would be of great value to have the reported activity progress data replace the original activity data. The reported progress information would then serve as the basis for calculating a revised project schedule. The implementation of this self-modifying capability would constitute a second valuable extension to PROJECT's progress control feature.

6.2 Some Concluding Remarks

It should be clear from the discussion in previous chapters that the computation of the status index is highly dependent on the activity progress data reported from the project site. Because of this dependence, the significance attachable to the index is bounded from above by how complete, up-to-date and accurate the reported activity progress data is.

The great amount of highly detailed cost and work progress data available to the project manager is clear evidence that much thought and effort has been expended in the area of project progress control. Perhaps because such a volume of information was available for the project manager, not many people seem to have considered the wisdom and, in fact, necessity of compacting the reams of progress data into a more tenable form--a form useful for pointing out those areas of the project requiring a deeper investigation. Among those who did recognize the value of this course of action was J. S. Baumgartner whose text PROJECT MANAGEMENT[3] provided the motivation for this work.

The paramount consideration in the development and implementation of the progress monitor has been to keep it simple--so the project manager can grasp it at a glance--yet meaningful. When, in the development of a feature like PROJECT's progress control capability, the emphasis is placed on simplicity and ease of understanding, sophistication must be sacrificed to a certain extent. The final point to be made, then, is

---

[3] Control and Management of Capital Projects, J. W. Hackney, John Wiley & Sons, Inc., New York, 1965, pg. 3.

that the status index should not be thought of as a panacea which will prevent a project from going awry. Rather, it is but an indicator and should never be a substitute for good judgment.

APPENDIX A

Appendix A consists of the programs which were written to implement PROJECT's progress control capability. The programs can be classified in two groups: (1) ICETRAN programs for computation and (2) programs written in the Command Definition Language, which serve to interpret the commands which the user submits.

The ICETRAN programs are grouped according to their respective load modules as follow:

Load Module PROGFL	Load Module PROGST	Load Module PROGTR
PROGFL	PROGST	PROGTR
PROGOL	PRPORK	PRGRAF
PROGIX	PRKDAT*	PRS85
PROGMD	PRLOGX*	PROGDJ
PRSAVE		
PROGET		

*The listing of this program is not included here since it is common to the entire PROJECT system and was not written specifically for the progress control feature.

ICETRAN LISTING

```
S.   1          SUBROUTINE PROGTR
S.   2          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.   3          COMMON NOHOL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   4          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,OBXXX(50)
S.   5          COMMON TABLE,HOLO,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.   6          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.   7          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.   8          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(0),
                1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.   9          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  10          DOUBLE PRECISION IALPHA,FMONTH,OBXXX
S.  11          EQUIVALENCE (INTXXX(17),ITIMRC)
S.  12          IF(K3)100,90,100
S.  13    C     ENTER HERE IF USER HAS SPECIFIED TIME RECOVERABLE DATA.
S.  14      100 IF(K5-2)1,2,3
S.  15        3 WRITE(6,4) NAME
S.  16        4 FORMAT(//,1X,'RESUBMIT TIME RECOVERABLE DATA FOR PROJECT ',A8,'.
                1LAST WORD OF COMMAND MUST BE  DAYS ,  WEEKS  OR LEFT BLANK.')
S.  17          ERROR RETURN
S.  18    C     TIME RECOVERABLE INPUT AS WEEKS -- CONVERT TO DAYS.
S.  19        2 ITIMRC=DNUM*K3
S.  20       10 WRITE(6,5) NAME, ITIMRC
S.  21        5 FORMAT(//,1X,'THE ESTIMATE OF TIME RECOVERABLE FOR PROJECT ',A8,'
                1IS ',I5,' DAYS.',/)
S.  22          RETURN
S.  23    C     TIME RECOVERABLE INPUT AS DAYS -- NO CONVERSION NEEDED.
S.  24        1 ITIMRC=K3
S.  25          GO TO 10
S.  26    C     ENTER HERE IF USER HASN'T SPECIFIED TIME RECOVERABLE DATA -- ASSUME TIME
S.  27    C     RECOVERABLE TO BE FIFTEEN PERCENT OF THE JOB DURATION.
S.  28       90 ITIMRC=((.15)*JOBDUR+.5)
S.  29          RETURN
S.  30          END
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PRGRAF
S.    2    C     PRGRAF PRINTS OUT COST & RESOURCE CURVES
S.    3    C     INPUT DATA IS IN NARROW, K5, RLXXX, INTXXX, & AB
S.    4          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.    5          COMMON NOHOL,JOBOUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.    6          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,AB(12),MAME,DBXXX(50)
S.    7          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.    8          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.    9          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.   10          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(0),
                1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.   11          DIMENSION PT(102),JOLD(11)
S.   12          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.   13          DOUBLE PRECISION IALPHA,FMONTH,DBXXX
S.   14          I1=LINE(1)
S.   15          I2=LINE(2)
S.   16          I3=LINE(3)
S.   17          I4=LINE(4)
S.   18          I5=LINE(5)
S.   19          K5=LINE(6)
S.   20          R1=I2-I1
S.   21          R2=I3-I1
S.   22          RRR=R1/R2
S.   23          I2=100*RRR
S.   24    C     ROUTINE TO REJECT 'Y' VALUES OUTSIDE THE RANGE 0. TO 2.0
S.   25          DO 3 I=1,K5
S.   26          IF(NARROW(I)-20000)1,1,2
S.   27    1     IF(NARROW(I))2,3,3
S.   28    C     OUTSIDE THE RANGE, PRINT ERROR MESSAGE
S.   29    2     IDAY=I1-1+I
S.   30          SIZE=NARROW(I)/10000.
S.   31          IF(NARROW(I))4,5,5
S.   32    4     NARROW(I)=0
S.   33          GO TO 3
S.   34    5     NARROW(I)=20000
S.   35    3     CONTINUE
S.   36    C
S.   37    C     SCALE ALL 'Y' VALUES
S.   38          Y=.0025
S.   39          DO 13 I=1,K5
S.   40   13     NARROW(I)=Y*NARROW(I)+.5
S.   41          J5=K5
S.   42    C
S.   43    C     ROUTINE TO MAKE A LINEAR FILL-IN IF FEWER THAN 100 POINTS INPUT
S.   44          IF(100-K5)14,70,70
S.   45       70 DEFINE MARROW,101,HALF,LOW
S.   46          JI=0
S.   47          DO 71 I=1,100
S.   48   71     MARROW(I)=0
```

```
S.  49          DO 75 I=1,K5
S.  50          II=(I-1)*100/(K5-1)+1.5
S.  51          MARROW(II)=NARROW(I)
S.  52          IF(JI)75,75,72
S.  53    C     SCALE VALUES IN MARROW BETWEEN THIS AND PREVIOUS POINT
S.  54    72    IX=JI+1
S.  55          JX=II-1
S.  56          IF(JX-IX)75,75,73
S.  57    73    LEN=JX-IX+1
S.  58          DO 74 J=1,LEN
S.  59          K=IX+J-1
S.  60    74    MARROW(K)=(MARROW(II)-MARROW(JI))*J/(II-JI)+MARROW(JI)+.5
S.  61    75    JI=II
S.  62          DESTROY NARROW
S.  63          DEFINE NARROW,101,HALF,LOW
S.  64          DO 76 I=1,100
S.  65    76    NARROW(I)=MARROW(I)
S.  66          DESTROY MARROW
S.  67          K5=100
S.  68    C
S.  69    C     LOOP TO PRINT OUT SUCCESSIVE LINES
S.  70    14    DO 50 KK=1,50
S.  71          DO 17 I=1,100
S.  72    17    PT(I)=' '
S.  73          DO 30 I=1,K5
S.  74          II=(I-1)*100/(K5-1)+1.5
S.  75          IF(100-II)18,19,19
S.  76       18 II=100
S.  77       19 CONTINUE
S.  78          IF(51-KK-NARROW(I))30,20,30
S.  79    20    IF(II-IZ)25,25,26
S.  80    25    PT(II)='*'
S.  81          GO TO 30
S.  82    26    PT(II)='-'
S.  83    30    CONTINUE
S.  84    31    Z=(51-KK)/5.
S.  85          IZ=Z
S.  86          IF(Z-IZ)35,40,35
S.  87    35     WRITE(6,36)(PT(I),I=1,100)
S.  88    36    FORMAT(10X,'.',100A1)
S.  89          GO TO 50
S.  90       40 W=(51-KK)/(Y*10000.)
S.  91          WRITE(6,41)W,(PT(I),I=1,100)
S.  92    41    FORMAT(1X,F3.2,'+',100A1)
S.  93    50    CONTINUE
S.  94          CALL PRS85(11,J5,I3)
S.  95          RETURN
S.  96          END
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PRS85(I1,J5,I3)
S.    2    C     THIS PRINTS OUT STATUS-GRAPH INDICES
S.    3          DIMENSION PT(103),JOLD(10)
S.    4          DO 1 I=1,100
S.    5    1     PT(I)='.'
S.    6          DO 6 I=1,J5
S.    7          II=(I-1)*100/(J5-1)+1.5
S.    8          IF(100-II)60,61,61
S.    9    60    II=100
S.   10    61    IF(100-J5)2,3,3
S.   11    2     XJ=II/10.
S.   12          GO TO 4
S.   13    3     IF(J5-33)32,31,31
S.   14    32    IF(J5-10)33,33,34
S.   15    33    XJ=I
S.   16          GO TO 4
S.   17    34    XJ=I/3
S.   18          GO TO 4
S.   19    31    XJ=I/10.
S.   20    4     JX=XJ
S.   21          IF(XJ-JX)6,5,6
S.   22    5     PT(II)='+'
S.   23    6     CONTINUE
S.   24          WRITE(6,63)(PT(J),J=1,100)
S.   25    63    FORMAT(6X,'0.0 ',100A1)
S.   26    C
S.   27    C     NOW, FIND WHERE '+' WAS PRINTED ON THE ABOVE LINE
S.   28          N=0
S.   29          XX='+'
S.   30          DO 8 I=1,100
S.   31          IF(XX-PT(I))8,7,8
S.   32    7     N=N+1
S.   33          JOLD(N)=I
S.   34    8     CONTINUE
S.   35          DO 9 I=1,102
S.   36    9     PT(I)=' '
S.   37    C     COMPUTE THE NEAREST INTEGER VALUE FOR THESE POINTS
S.   38          IF(N)90,90,101
S.   39    101   ADD=(I3-I1)/(N-1)
S.   40          DO 30 I=1,N
S.   41          JHOLD=I1+(I-1)*ADD+.5
S.   42    10    J3=JHOLD/100
S.   43          IF(J3)11,12,11
S.   44    11    K=JOLD(I)-1
S.   45          KK=1
S.   46          GO TO (21,22,23,24,25,26,27,28,29),J3
S.   47    12    J2=JHOLD/10-10*J3
S.   48          K=JOLD(I)
S.   49          KK=2
```

```
S.  50          IF(J2)14,13,14
S.  51    13    IF(J3)20,15,20
S.  52    14    GO TO (21,22,23,24,25,26,27,28,29),J2
S.  53    15    J1=JHOLD-10*J2-100*J3
S.  54          KK=3
S.  55          K=JOLD(I)+1
S.  56          IF(J1)16,151,16
S.  57    151   IF(J2)20,152,20
S.  58    152   IF(J3)20,30,20
S.  59    16    GO TO (21,22,23,24,25,26,27,28,29),J1
S.  60    20    PT(K)='0'
S.  61          GO TO (12,15,30),KK
S.  62    21    PT(K)='1'
S.  63          GO TO (12,15,30),KK
S.  64    22    PT(K)='2'
S.  65          GO TO (12,15,30),KK
S.  66    23    PT(K)='3'
S.  67          GO TO (12,15,30),KK
S.  68    24    PT(K)='4'
S.  69          GO TO (12,15,30),KK
S.  70    25    PT(K)='5'
S.  71          GO TO (12,15,30),KK
S.  72    26    PT(K)='6'
S.  73          GO TO (12,15,30),KK
S.  74    27    PT(K)='7'
S.  75          GO TO (12,15,30),KK
S.  76    28    PT(K)='8'
S.  77          GO TO (12,15,30),KK
S.  78    29    PT(K)='9'
S.  79          GO TO (12,15,30),KK
S.  80    30    CONTINUE
S.  81          WRITE(6,40)(PT(J),J=1,102)
S.  82    40    FORMAT(9X,102A1,' DAYS')
S.  83          RETURN
S.  84    90    WRITE(6,91)N
S.  85    91    FORMAT(/5X,' ERRORS IN INDEXING, N = ',I2)
S.  86          RETURN
S.  87          END
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PROGDJ
S.    2          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.    3          COMMON NOHOL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.    4          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,DBXXX(50)
S.    5          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.    6          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.    7          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.    8          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(D),
                1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.    9          DYNAMIC ARRAY KPROG
S.   10          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.   11          DOUBLE PRECISION IALPHA,FMONTH,DBXXX
S.   12    P     EQUIVALENCE (KPROG(1),OBXXX(22))
S.   13    C     THIS SUBROUTINE IS EXECUTED FROM THE COL SELINX.  ITS PURPOSE IS TO
S.   14    C     DESTROY THE ARRAY JSORT AFTER ALL COMPUTATIONS HAVE BEEN MADE.
S.   15          DESTROY JSORT
S.   16          DESTROY KPROG
S.   17          RETURN
S.   18          END
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PROGST
S.    2          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.    3          COMMON NOHOL,JOBOUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.    4          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,D8XXX(50)
S.    5          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.    6          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.    7          COMMON SORT(P),JSORT(P),KHOLD(P),MOUM(P),MARROW(P),ICAL(P),IHOL(P)
S.    8          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(D),
                1JSORT,KHOLD,MOUM,MARROW,ICAL,IHOL
S.    9          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.   10          DOUBLE PRECISION IALPHA,FMONTH,D8XXX
S.   11          DOUBLE PRECISION NAME8
S.   12          DYNAMIC ARRAY IDPROG
S.   13          EQUIVALENCE (D8XXX(9),NAME8)
S.   14          EQUIVALENCE (INTXXX(15),I2)
S.   15          EQUIVALENCE (INTXXX(10),(PROG)
S.   16          NAME8=NAME
S.   17          CALL ADDNUM(NAME8,8)
S.   18          GO TO (10,20,30,40,50,60,70),K1
S.   19    C     SEE IF NAME8 EXISTS.  IF YES, PULL IDPROGS.
S.   20       10 LEN=4*NOACT
S.   21          DISK FILE INFO(2,NAME8,IFSTAT,IH,IL)
S.   22          GO TO (11,13,13),IFSTAT
S.   23    C     CREATE THE FILE IF NONE EXISTS FOR THIS PROJECT.
S.   24       11 DO 12 I=1,10
S.   25       12 LINE(I)=0
S.   26          DISK OPEN(2,NAME8,1)
S.   27    C     IH IS THE IDENTIFIER OF RECORD ONE OF NAME8.
S.   28          IH=0
S.   29          DISK PUT(2,NAME8,IH,LINE(1),1,40)
S.   30    C     ZERO IN RECORD OF IDPROG'S.
S.   31          DESTROY IDPROG
S.   32          DEFINE IDPROG,NOACT,FULL,LOW
S.   33    C     II IS IDENTIFIER OF OF IDPROG RECORD.
S.   34          II=0
S.   35          DISK PUT(2,NAME8,II,IDPROG(I),1,LEN)
S.   36    C     STORE IDENTIFIER OF IDPROG RECORD IN 1ST WORD OF 1ST RECORD OF NAME8.
S.   37          LINE(1)=II
S.   38          DISK PUT(2,NAME8,IH,LINE(1),1,4)
S.   39          GO TO 14
S.   40    C     FILE DOES EXIST FOR THIS PROJECT.
S.   41       13 DISK OPEN(2,NAME8,1)
S.   42          DISK GET(2,(H,LINE(1),D,0,JUNK,NEXT)
S.   43          II=LINE(1)
S.   44          DEFINE (DPROG,NOACT,FULL,LOW
S.   45          DISK GET(2,(I,IDPROG(1),0,0,JUNK,NEXT)
S.   46    C     NAME8 EXISTS.  STORE DATE OF REPORT IN THE STATUS WORD.
S.   47       14 (PROG=0
S.   48    C     PUT THE ACTIVITY NUMBERS IN A LIST.
```

```
S.  49          KI=1
S.  50          CALL RETRVE
S.  51          RETURN
S.  52    C     ************************************************************************
S.  53    C     ENTER HERE WITH AN ACTIVITY REPORTING PROGRESS.
S.  54    C     PROGRESS DATA ON COSTS GOES IN 1ST WORD OF NAME8, FINISH INFO GOES IN
S.  55    C     SECOND WORD, REPORTED START DATA GOES IN WORD 3 OF NAME8.
S.  56       20 IA=A
S.  57          IB=B
S.  58    C     CALL PRLOGX TO SEARCH THE LIST FOR THE ACTIVITY WE SEEK.
S.  59          CALL PRLOGX(IA,IB,K3,INGRID,NOACT)
S.  60    C     IF K3 IS POSITIVE, K3 REPRESENTS ITS POSITION IN THE LIST.  IF IT
S.  61    C     IS NEGATIVE, THE ACTIVITY IS NOT IN A PART OF THE NETWORK.  IN THE
S.  62    C     SECOND CASE, PRINT AN ERROR MESSAGE.
S.  63          IF(K3)21,21,22
S.  64       21 IF(MODE)27,27,24
S.  65       27 WRITE(6,28) IA
S.  66       28 FORMAT(//,1X,'PROGRESS DATA ON ACTIVITY ',I4,' IS NOT ACCEPTED SIN
                2CE THIS ACTIVITY IS NOT A PART OF THE NETWORK.',/)
S.  67          RETURN
S.  68       24 WRITE(6,29) IA,IB
S.  69       29 FORMAT(//,1X,'ACTIVITY ',I4,' - ',I4,' IS NOT A PART OF THE NETWOR
                2K.  PROGRESS DATA FOR THIS ACTIVITY IS NOT ACCEPTED.',/)
S.  70          RETURN
S.  71       22 IF(IDPROG(K3))200,200,201
S.  72    C     IDPROG POSITIVE IMPLIES SOME DATA HAS ALREADY BEEN REPORTED.
S.  73      201 DISK GET(2,IDPROG(K3),LINE(1),1,12)
S.  74          GO TO 204
S.  75      200 DO 23 J=1,9
S.  76       23 LINE(J)=0
S.  77      204 RETURN
S.  78    C     ************************************************************************
S.  79    C     STORE REPORTED FINISH DATE.
S.  80    C     IF REPORT DATE IS AFTER A REPORTED FINISH DATE (INDICATING THAT AN ACT.
S.  81    C     IS, IN FACT, FINISHED),STORE THE DATE AS REPORTED.  TO INDICATE AN EST.
S.  82    C     FINISH DATE (I.E., FINISH DATE IS LATER THAN THE REPORT DATE) STORE THE
S.  83    C     NEGATIVE OF THE DATE REPORTED.
S.  84       30 IF(IPROG-D)35,31,31
S.  85       31 LINE(2)=D
S.  86          GO TO 36
S.  87       35 LINE(2)=-D
S.  88       36 CONTINUE
S.  89          RETURN
S.  90    C     ************************************************************************
S.  91    C     STORE REPORTED START DATE.
S.  92       40 LINE(3)=D
S.  93          RETURN
S.  94    C     ************************************************************************
S.  95    C     STORE REPORTED COST INFORMATION.
S.  96       50 LINE(1)=C
S.  97          RETURN
S.  98    C     ************************************************************************
S.  99       60 IF(I2-)37,83,37
S. 100       83 IF(LINE(3))81,37,81
S. 101       81 IF(IABS(LINE(2)))80,37,80
```

```
S. 102       80 IF(LINE(3)-IABS(LINE(2)))37,37,38
S. 103    C     SEE IF FINISH IS SMALLER THAN THE START.
S. 104       38 IF(MODE)381,381,382
S. 105      381 WRITE(6,383) IA,LINE(3),LINE(2)
S. 106      383 FORMAT(/,1X,'ACTIVITY ',I5,' REPORTS START ON DAY ',I5,' FINISH ON
               1 DAY ',I5)
S. 107          GO TO 390
S. 108      382 WRITE(6,384) IA,IB,LINE(2),LINE(3)
S. 109      384 FORMAT(/,1X,'ACTIVITY ',I4,' - ',I4,' REPORTS START ON DAY ',I5,'
               1FINISH ON DAY ',I5)
S. 110      390 WRITE(6,391)
S. 111      391 FORMAT(1X,'***ERROR -- FINISH CANNOT OCCUR BEFORE START.  ALL STAR
               IT AND FINISH DATA FOR THIS ACTIVITY HAVE BEEN ZEROED.')
S. 112          LINE(2)=0
S. 113          LINE(3)=0
S. 114       37 DISK PUT(2,NAME8,IDPROG(K3),LINE(1),1,36)
S. 115          RETURN
S. 116    C     ********************************************************************
S. 117       70 DISK OPEN(2,NAME8,I)
S. 118          DISK PUT(2,NAME8,II,IDPROG(1),1,LEN)
S. 119          DISK CLOSE(2,NAME8)
S. 120          DESTROY IDPROG
S. 121          RETURN
S. 122          END
```

ICETRAN LISTING

```
S.   1          SUBROUTINE PRPORK
S.   2    C     PRWORK TAKES A DATE (MO,DY,YR IN A,B,C) AND PUTS WORK DAY INTO D
S.   3    C     THIS EDITION OF PRPORK MODIFIED 27 JULY 67 BY J.T. MACDERMOTT TO DO ERROR
S.   4    C     RETURNS IF ANY PART OF THE GIVEN DATE IS INVALID.  THIS DONE SO AS TO
S.   5    C     INSURE PROPER EXECUTION OF SUBSEQUENT COMMANDS.
S.   6          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.   7          COMMON NOHOL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   8          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,OBXXX(50)
S.   9          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.  10          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.  11          COMMON SORT(P),JSORT(P),KHOLD(P),MOUM(P),MARROW(P),ICAL(P),IHOL(P)
S.  12          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IOTIME,IDNET,SORT(0),
                1JSORT,KHOLD,MOUM,MARROW,ICAL,IHOL
S.  13          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  14          DOUBLE PRECISION IALPHA,FMONTH,OBXXX
S.  15          DOUBLE PRECISION MON
S.  16    C     SET 'D' = 0 IN CASE NO VALID WORK DAY CAN BE FOUND (IF GIVEN DATE BAD)
S.  17          D=0
S.  18    C     HOLD EXISTING JOBDUR ASIDE IN CASE PSEUDO JOBDUR IS USED TO GET CALENDAR
S.  19          MOBDUR=JOBDUR
S.  20    C     PUT A VALUE INTO JOBDUR IN CASE A CALENDAR MUST BE GENERATED
S.  21          JOBDUR=300
S.  22    C     SEE IF A STARTING DATE HAS BEEN ASSIGNED
S.  23      200 IF(DATEM)510,510,515
S.  24    C     NO, PRINT ERROR MESSAGE
S.  25      510 WRITE(6,900)
S.  26          GO TO 100
S.  27      515 INDATM=A
S.  28          INDATD=B
S.  29          INDATY=C
S.  30    C     SEE IF YEAR WAS GIVEN, AS IT MUST BE
S.  31          IF(INDATY)300,300,505
S.  32      300 WRITE(6,901)
S.  33          GO TO 100
S.  34    C     SEE IF GIVEN DATE OCCURS BEFORE PROJECT START
S.  35      505 IF(10000.*DATY+100.*DATEM+DATED-10000.*C-100.*A-B)506,506,504
S.  36      504 J=DATED
S.  37          K=DATEM
S.  38          L=DATEY+1900
S.  39          MON=FMONTH(K)
S.  40          BLANK=FMONTH(INDATM)
S.  41          WRITE(6,902)INDATD,BLANK,INDATY,J,MON,L
S.  42          GO TO 100
S.  43    C     DESTROY ANY ICAL LEFT AROUND
S.  44      506 DESTROY ICAL
S.  45    C     YES, GENERATE A CALENDAR
S.  46      507 CALL DATE
S.  47    C     MATCH REPORT DAY TO CALENDAR TO FIND CORRESPONDING JOB DAY
S.  48    C     MATCH THE YEAR
```

```
S.   49          DO 520 I=1,JOBDUR
S.   50          IF(ICAL(3,I)-INDATY)520,525,505
S.   51    525   INDAY=I
S.   52          GO TO 530
S.   53    520   CONTINUE
S.   54          JOBDUR=JOBDUR+200
S.   55          GO TO 507
S.   56    C     MATCH THE MONTH
S.   57    530   DO 540 I=INDAY,JOBDUR
S.   58          IF(ICAL(1,I)-INDATM)540,535,537
S.   59    C     SEE IF THIS IS THE SAME YEAR STILL
S.   60      535 IF(INDATY-ICAL(3,I))538,536,536
S.   61      538 INDAY=I-1
S.   62          GO TO 545
S.   63      536 INDAY=I
S.   64          GO TO 545
S.   65    C     GIVEN MONTH HAS NO PROJECT WORK DAYS. PRINT MESSAGE AND RETURN
S.   66      537 BLANK=FMONTH(INDATM)
S.   67          INDATY=INDATY+1900
S.   68          WRITE(6,903) INDATD,BLANK,INDATY,NAME
S.   69          GO TO 100
S.   70    540   CONTINUE
S.   71    C     IF CONTROL COMES HERE, MONTH NUMBER NOT FOUND IN ICAL IN YEAR
S.   72          JOBDUR=JOBDUR+200
S.   73          GO TO 507
S.   74    C     MATCH THE DAY
S.   75    545   DO 555 I=INDAY,JOBDUR
S.   76          IF(ICAL(2,I)-INDATD)552,551,550
S.   77    C     GIVEN DATE NOT A WORKING DAY, BACK UP ONE
S.   78    550   INDAY=I-1
S.   79          K5=1
S.   80    C     CHECK TO SEE IF DAY OF REPORT OCCURS BEFORE PROJECT
S.   81      556 IF(INDAY)504,504,500
S.   82    C     THE MATCH FOUND, NOW RETURN
S.   83      551 INDAY=I
S.   84          K5=0
S.   85          GO TO 500
S.   86    552   IF(ICAL(1,I)-INDATM)555,555,550
S.   87    555   CONTINUE
S.   88    C     DAY NOT FOUND IN CALENDAR, GENERATE A LONGER ONE
S.   89          JOBDUR=JOBDUR+30
S.   90          GO TO 507
S.   91    C     INDAY IS DAY TO ASSIGN
S.   92      500 D=INDAY
S.   93    C     PUT JOBDUR BACK AS WAS AND RETURN
S.   94          JOBDUR=MOBDUR
S.   95          RETURN
S.   96      100 JOBDUR=MOBDUR
S.   97          ERROR RETURN
S.   98      900 FORMAT(//' ******'/,' A PROJECT START DATE MUST BE ASSIGNED BEFORE
                 1 OTHER START AND FINISH DATES MAY BE ASSIGNED'/,' EITHER WORK WITH
                 2OUT DATES, OR FIRST ASSIGN A PROJECT START DATE.'/,' ******'//)
S.   99      901 FORMAT(//' ******'/' A START OR FINISH OCCURRING BEFORE PROJECT ST
                 1ART HAS BEEN ASSIGNED.'/,' THIS IS NOT ALLOWED.'//)
S.  100      902 FORMAT(//' THE GIVEN DATE',I3,1X,A3,I5,' OCCURS BEFORE PROJECT STA
```

```
     1RT',I3,1X,A3,I5,', AND SO IT MAY NOT BE USED.'/,' NO ACTION TAKEN.
     2'/)
S. 101    903 FORMAT(//' ******'/,' THE MONTH OF THE GIVEN DATE',I3,1X,A3,I5,' I
     1NCLUDES NO WORKING DAYS FOR PROJECT ''',A8,'''.'/,' NO ACTION TAKEN
     2N.'/)
S. 102       END
```

ICETRAN LISTING

```
S.   1          SUBROUTINE PROGFL(JBOR)
S.   2    C     FLOW IS THE CALLING SUBROUTINE FOR SCHEDULING COMPUTATIONS
S.   3    C     SUBROUTINE FLOW REVISED BY T. MACDERMOTT 14 JUNE 1967.  CHANGE IS THAT
S.   4    C     NOW PROGRAM IS CALLED PROGFL -- AND IT CALLS PROGSF (STMT 52) VICE PRSFEX.
S.   5          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,DNUM
S.   6          COMMON NOHOL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   7          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,AB(12),MAME,DBXXX(50)
S.   8          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.   9          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IOTIME(P),IDNET(P)
S.  10          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.  11          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IOTIME,IDNET,SORT(0),
                1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.  12          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  13          DOUBLE PRECISION IALPHA,FMONTH,OBXXX
S.  14    C     DESTROY CALENDAR IN ICAL WHEN CALCULATING NEW SCHEDULE
S.  15          DESTROY ICAL
S.  16          K1=9
S.  17          CALL RETRVE
S.  18    C     RECOVER NARROW FROM SECONDARY STORAGE
S.  19    C     DESTROY NARROW FIRST TO BE SURE THE AS INPUT IS NOT USED
S.  20          DESTROY NARROW
S.  21          K1=5
S.  22          CALL RETRVE
S.  23    C     PUT KT AT TOP OF LIST BEING CONSIDERED, KB AT BOTTOM. THESE MAY CHANGE
S.  24          I=-1
S.  25       28 KT=I+2
S.  26          KB=NUMAR
S.  27    C     USE APPROPRIATE DURATIONS WITH GIVEN LAG VALUE TO SET UP RELATIONSHIP
S.  28          DO 39 I=KT,KB
S.  29    C     IS THERE A LAG OF ANY KIND
S.  30          IF(NARROW(3,I))30,39,30
S.  31    C     YES, GET THE OPERATOR, PLUS 1, TO BRANCH ON
S.  32    C     GET THE INPUT LAG VALUE
S.  33       30 KN=NARROW(3,I)
S.  34          K=IABS(KN-(KN/10)*10)+1
S.  35          KN=KN/10
S.  36    C     NOW BRANCH ON OPERATOR
S.  37          GO TO(34,31,32,33),K
S.  38    C     START-START, SUBTRACT DURATION OF INDEPENDENT ACTIVITY, ADD LAG VALUE
S.  39       31 NARROW(3,I)=KN-KTIME(1,NARROW(1,I))
S.  40    C     GET THE PROPER LAG FOR THE NEW ARROW
S.  41          KN=-KN-KTIME(1,NARROW(2,I))
S.  42    C     NOW GO AND ADD NEW ARROW
S.  43          GO TO 40
S.  44    C     START-FINISH, SUBTRACT BOTH DURATIONS AND LAG FACTOR
S.  45       32 NARROW(3,I)= KN-KTIME(1,NARROW(1,I))-KTIME(1,NARROW(2,I))
S.  46    C     THE NEW ARROW HAS THE SAME KN, BUT WITH THE OTHER ALGEBTAIC RIGN
S.  47          KN=-KN
S.  48    C     NOW GO AND ADD ANOTHER ARROW
```

```
S.   49          GO TO 40
S.   50    C     FINISH-FINISH, SUBTRACT DURATION OF DEPENDENT ACTIVITY , ADD LAG VALUE
S.   51       33 NARROW(3,I)=KN-KTIME(1,NARROW(2,I))
S.   52    C     GET THE PROPER LAG FOR THE NEW ARROW
S.   53          KN=-KN-KTIME(1,NARROW(1,I))
S.   54    C     NOW GO AND ADD ANOTHER ARROW FOR THIS F-F RELATIONSHIP
S.   55          GO TO 40
S.   56       34 NARROW(3,I)=KN
S.   57       39 CONTINUE
S.   58          GO TO 52
S.   59    C     FOR S-S,S-F, AND F-F RELATIONSHIPS ANOTHER ARROW IS ADDED TO TOP OF LIST
S.   60    C     FIRST INCREMENT NUMAR, THEN REDEFINE NARROW ONE LONGER
S.   61       40 NUMAR=NUMAR+1
S.   62          DEFINE NARROW,3,NUMAR,HALF,LOW
S.   63    C     NOW SHIFT ALL OF NARROW DOWN BY ONE
S.   64          DO 310 KK=2,NUMAR
S.   65          KJ=NUMAR-KK+1
S.   66          DO 310 KM=1,3
S.   67      310 NARROW(KM,KJ+1)=NARROW(KM,KJ)
S.   68    C     PUT NEW (PSEUDO) ARROW ON TOP OF LIST
S.   69          NARROW(1,1)=NARROW(2,I+1)
S.   70          NARROW(2,1)=NARROW(1,I+1)
S.   71          NARROW(3,1)=KN
S.   72    C     IF THIS WAS LAST ARROW, LEAVE LOOP AND GO ON TO 40. IF THERE ARE MORE
S.   73    C     ARROWS, THEN GO BACK AND START LOOP AGAIN WITH NEXT ARROW, NOW 2 BELOW.
S.   74          IF(I+1-NUMAR)28,52,52
S.   75    C     FLOW NETWORK TO FIND ALL EARLY STARTS AND FINISHES
S.   76    C     CALL PROGFF TO PERFORM FORWARD FLOW -- THIS CALL PROGBF WHICH DOES THE
S.   77    C     BACKWARD FLOWING.
S.   78       52 CALL PROGOL
S.   79          DESTROY NARROW
S.   80    C     BRING STATUS WORDS UP TO DATE, HOLD JOBOUR SO IT WON'T BE RUN OVER
S.   81          JBOR=JOBOUR
S.   82    C     GET BACK STATUS WORDS IN CASE NUMAR WAS CHANGED
S.   83          K1=10
S.   84          CALL RETRVE
S.   85          RETURN
S.   86          END
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PROGIX
S.    2          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLDW,DATEM,DATED,DATEY,DNUM
S.    3          COMMON NOHDL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.    4          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,AB(12),MAME,DBXXX(50)
S.    5          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.    6          COMMON NARROW(P),(NCODE(P),INGR(D(P),KTIME(P),IDTIME(P),IDNET(P)
S.    7          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.    8          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(D),
                1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.    9          DYNAMIC ARRAY KPROG, IDPROG, JCOST, IACTRC
S.   10          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.   11          DOUBLE PRECISION IALPHA,FMONTH,DBXXX
S.   12          DOUBLE PRECISION NAME7
S.   13          INTEGER F
S.   14          REAL RLXXX
S.   15          EQUIVALENCE (DBXXX(8),NAME7)
S.   16          EQUIVALENCE (INTXXX(6),NSEL)
S.   17          EQUIVALENCE ((NTXXX(17),IT(MRC)
S.   18          EQUIVALENCE (NAME8,DBXXX(9))
S.   19          EQUIVALENCE ((NTXXX(14),(1)
S.   20          EQUIVALENCE (RLXXX(17),WP),(RLXXX(19),CI)
S.   21    P     EQUIVALENCE ((DPROG(1),DBXXX(23))
S.   22    P     EQUIVALENCE (KPROG(I),DBXXX(22))
S.   23          KKK=INTXXX(20)
S.   24    C     SAVE THE WORKDAY SET IN D BY THE CDL.
S.   25          IWKDAY=D
S.   26          JWKDAY=IWKDAY
S.   27          DISK FILE INFO(2,NAME8,IFSTAT,IH,IL)
S.   28          IF(IFSTAT-1)3,3,4
S.   29        3 WRITE(6,9)
S.   30        9 FORMAT(/,1X,'NO PROGRESS DATA HAS BEEN REPORTED.  NO COMPUTATIONS
                1CAN BE MADE.')
S.   31          ERROR RETURN
S.   32        4 IF(ISTAT(JUNK,JSORT)-I)69,69,65
S.   33    C     ENTRY HERE IMPLIES JSORT MUST BE CREATED.
S.   34       65 DEFINE JSORT,NOACT,HALF,LOW
S.   35          DO 70 I=I,NOACT
S.   36       70 JSORT(I)=(
S.   37          NSEL=NOACT
S.   38    C     ENTRY HERE IMPLIES JSORT EXISTS.
S.   39    C     INITIALIZE THE KTIME ARRAY.
S.   40       69 DESTROY KTIME
S.   41          DEFINE KTIME,7,POINTER,LOW
S.   42          DO 5 I=1,7
S.   43          DEFINE KTIME(I),NOACT,HALF,LOW
S.   44        5 RELEASE KTIME(I)
S.   45          DO 6 (=1,NSEL
S.   46          J=JSORT(I)
S.   47          IG=IDNET(J)
```

```
S.  48          DISK GET(2,IG,LINE(1),9,12)
S.  49        6 KTIME(1,J)=LINE(1)
S.  50          RELEASE IONET
S.  51    C     FIRST CALL TO PROGFL RETURNS PLANNED PHASE DURATION.
S.  52          IF(NOACT-NSEL)401,402,401
S.  53      401 CALL PROGFL(JBOR)
S.  54          GO TO 403
S.  55      402 JBOR=JOBDUR
S.  56      403 ORGDUR=JBOR-1
S.  57    C     CALL TO PROGET RETRIEVES PROGRESS DATA STORED IN NAME 8.
S.  58          CALL PROGET(1,1,3)
S.  59          IF(NOACT-NSEL)34,35,34
S.  60       35 DO 30 I=1,NSEL
S.  61          JJ=JSORT(I)
S.  62          IF(IDPROG(JJ))30,30,31
S.  63       31 KPR=IABS(KPROG(2,JJ))
S.  64          IF(KPR)501,501,502
S.  65    C     NO FINISH REPORTED.
S.  66      501 IF(KPROG(3,JJ))503,503,504
S.  67    C     NO START EITHER.
S.  68      503 KSWTCH=1
S.  69          GO TO 550
S.  70    C     NO FINISH BUT A START
S.  71      504 KSWTCH=2
S.  72          GO TO 550
S.  73    C     A FINISH REPORTED -- HOW ABOUT A START
S.  74      502 IF(KPROG(3,JJ))505,505,506
S.  75    C     FINISH BUT NO START.
S.  76      505 KSWTCH=3
S.  77          GO TO 550
S.  78    C     START AND A FINSIH.
S.  79      506 KSWTCH=4
S.  80      550 GO TO (30,300,33,36),KSWTCH
S.  81    C     START ONLY
S.  82      300 KTIME(2,JJ)=KPROG(3,JJ)
S.  83          KTIME(3,JJ)=KPROG(3,JJ)
S.  84          KTIME(4,JJ)=KPROG(3,JJ)+KTIME(1,JJ)-1
S.  85          KTIME(5,JJ)=KPROG(3,JJ)+KTIME(1,JJ)-1
S.  86          GO TO 30
S.  87       33 KTIME(2,JJ)=KPR-KTIME(1,JJ)+1
S.  88          KTIME(3,JJ)=KTIME(2,JJ)
S.  89          KTIME(4,JJ)=KPR
S.  90          KTIME(5,JJ)=KPR
S.  91          GO TO 30
S.  92    C     IF BOTH S AND F ARE GIVEN, SET KTIME ES'S, LS'S, EF'S, LF'S.
S.  93       36 KTIME(2,JJ)=KPROG(3,JJ)
S.  94          KTIME(3,JJ)=KPROG(3,JJ)
S.  95          KTIME(4,JJ)=KPR
S.  96          KTIME(5,JJ)=KPR
S.  97          KTIME(1,JJ)=KPR-KPROG(3,JJ)+1
S.  98       30 CONTINUE
S.  99          GO TO 370
S. 100       34 DO 37 I=1,NSEL
S. 101          J=JSORT(I)
S. 102          KPR=IABS(KPROG(2,J))
```

```
S. 103          IF(KPR)37,37,38
S. 104       38 IF(KPROG(3,J))37,37,39
S. 105       39 KTIME(1,J)=KPR-KTIME(3,J)+1
S. 106       37 CONTINUE
S. 107    C     CHECK TO SEE WHICH INDEX IS REQUESTED.  K3=1 IMPLIES COST INDEX ONLY, K3=
S. 108    C     2 IMPLIES WORK PROGRESS INDEX ONLY, K3=3 IMPLIES STATUS INDEX.
S. 109      370 DO 40 J=1,NSEL
S. 110          I=JSORT(J)
S. 111    C     SEE IF A FINISH HAS BEEN REPORTED.
S. 112          IF(KPROG(2,I))41,42,41
S. 113       42 KTIME(4,I)=0
S. 114          KTIME(5,I)=0
S. 115       41 IF(KPROG(3,I))40,43,40
S. 116       43 KTIME(3,I)=0
S. 117          KTIME(2,I)=0
S. 118       40 CONTINUE
S. 119    C     SECOND CALL TO PROGFL RETURNS LATEST REVISED ESTIMATE OF PHASE/PROJECT
S. 120          CALL PROGFL(JBDR)
S. 121    C     CHECK THE VALUE OF ICOST TO SEE IF ESTIMATED COST DATA EXISTS IN NAME7.
S. 122          K1=10
S. 123          CALL RETRVE
S. 124          ICOST=INTXXX(9)
S. 125          IF(INTXXX(20)-2)200,72,200
S. 126      200 IF(ICOST)71,71,72
S. 127    C     IF ICOST IS ZERO, PRINT ERROR MESSAGE AND RETURN -- OTHERWISE, GO AHEAD.
S. 128       71 WRITE(6,710) NAME
S. 129      710 FORMAT(//,1X,'COST DATA HAS NOT BEEN FILED FOR PROJECT ',A8,'.  CO
                 IST INDEX CANNOT BE COMPUTED.',/)
S. 130          DESTROY JSORT
S. 131          DESTROY KPROG
S. 132          ERROR RETURN
S. 133       72 NAME7=NAME
S. 134          CALL ADDNUM(NAME7,7)
S. 135          DISK FILE INFO(2,NAME7,JUNK,IG,IL)
S. 136          DISK GET(2,IG,LINE(1),0,8,JUNK,NEXT)
S. 137    C     COMPUTE COST COMPONENT OF STATUS INDES.  RETRIEVE EST. COST DATA FROM
S. 138    C     NAME7.
S. 139          IG=LINE(1)
S. 140          LEN=LINE(2)
S. 141          DEFINE IACTRC,LEN,FULL,LOW
S. 142          DISK GET(2,IG,IACTRC(1),0,0,JUNK,NEXT)
S. 143          DEFINE JCOST,LEN,FULL,LOW
S. 144          DO 73 I=1,LEN
S. 145          IG=IACTRC(I)
S. 146          DISK GET(2,IG,LINE(1),0,4,JUNK,NEXT)
S. 147       73 JCOST(I)=LINE(1)/100
S. 148          DESTROY IACTRC
S. 149          IACTCO=0
S. 150          IESTCO=0
S. 151    C     COMPUTE ESTIMATED COST TO DATE.
S. 152    C     ALWAYS US JCOST WHEN COMPUTING ESTIMATED COSTS.  DON'T EVER USE KPROG
S. 153    C     COSTS.  IF ACTIVITY IS FINISHED, TAKE WHOLE COST -- IF IN PROGRESS, TAKE
S. 154    C     LINEAR PORTION OF COST -- IF NOT STARTED, TAKE ZERO COST.
S. 155          IF(NOACT-NSEL)606,605,606
S. 156      606 DESTROY IDNET
```

```
S. 157          DESTROY 10TIME
S. 158          K1=9
S. 159          CALL RETRVE
S. 160          DO 600 I=1,NSEL
S. 161          J=JSORT(1)
S. 162          IG=10TIME(J)
S. 163          DISK GET(2,IG,LINE(2),5,20)
S. 164          DO 601 L=2,5
S. 165        601 KTIME(L,J)=LINE(L)
S. 166        600 CONTINUE
S. 167          DO 607 I=1,NSEL
S. 168          J=JSORT(I)
S. 169          KPR=IABS(KPROG(2,J))
S. 170          IF(KPR)609,609,610
S. 171        610 KTIME(4,J)=KPR
S. 172          KTIME(5,J)=KPR
S. 173        609 IF(KPROG(3,J))607,607,611
S. 174        611 KTIME(2,J)=KPROG(3,J)
S. 175          KTIME(3,J)=KPROG(3,J)
S. 176        607 CONTINUE
S. 177        605 DO 74 I=1,NSEL
S. 178          J=JSORT(I)
S. 179          IF(IWKDAY-KTIME(2,J))74,76,76
S. 180         76 IF(IWKDAY-KTIME(4,J))75,77,77
S. 181         77 IESTCO=IESTCO+JCOST(J)
S. 182          GO TO 74
S. 183         75 DIFF=IWKDAY-KTIME(2,J)
S. 184          DIV=KTIME(1,J)
S. 185          IESTCO=IESTCO+((DIFF/DIV)*JCOST(J))
S. 186         74 CONTINUE
S. 187    C     COMPUTE ACTUAL COST TO DATE.
S. 188    C     IF ACTIVITY IS IN PROGRESS, TAKE LINEAR PORTION OF COST -- IF FINISHED,
S. 189    C     TAKE ENTIRE COST AS REPORTED IN NAME8 (OR NAME7 IF NOTHING IN NAME8) --
S. 190    C     IF NOT STARTED ASSIGN ZERO COST.
S. 191          DO 80 I=1,NSEL
S. 192          J=JSORT(I)
S. 193          IF(IWKDAY-KTIME(2,J))80,82,82
S. 194         82 IF(IWKDAY-KTIME(4,J))81,83,83
S. 195         83 IF(KPROG(I,J))84,84,85
S. 196         84 IACTCO=IACTCO+JCOST(J)
S. 197          GO TO 80
S. 198         85 IACTCO=IACTCO+KPROG(I,J)
S. 199          GO TO 80
S. 200         81 DIFF=(WKDAY-KTIME(2,J)
S. 201          DIV=KTIME(I,J)
S. 202          IF(KPROG(I,J))86,86,87
S. 203         86 IACTCO=IACTCO+((DIFF/DIV)*JCOST(J))
S. 204          GO TO 80
S. 205         87 IACTCO=IACTCO+((DIFF/DIV)*KPROG(I,J))
S. 206         80 CONTINUE
S. 207          IPHCO=0
S. 208          DO 90 I=1,NSEL
S. 209          J=JSORT(I)
S. 210    C     COMPUTE ESTIMATED PHASE OR PROJECT COST.
S. 211         90 IPHCO=IPHCO+JCOST(J)
```

```
S. 212          DESTROY JCOST
S. 213    C     COMPUTE THE COST INDEX.
S. 214          ACOST=IACTCO
S. 215          ECOST=IESTCO
S. 216          TCOST=IPHCO
S. 217          CI=(TCOST-(ACOST-ECOST))/TCOST
S. 218    C     IF NO START AND/OR FINISH DATA HAS BEEN REPORTED, ZERO THE APPROPRIATE
S. 219    C     PLACES IN THE KTIME ARRAY.
S. 220          DESTROY KPROG
S. 221    C     SEE IF ITIMRC HAS BEEN DEFINED -- IF NOT, MAKE IT 15 PERCENT OF JOBDUR.
S. 222          IF(ITIMRC)737,786,737
S. 223      786 ITIMRC=((.15)*JOBDUR)+.5
S. 224          WRITE(6,787) ITIMRC
S. 225      787 FORMAT(/,1X,'ASSIGNMENT OF TIME RECOVERABLE WAS NOT MADE.  IT IS T
               1AKEN TO BE ',I4,' DAYS.')
S. 226    C     TN=TIME NOW.
S. 227      737 TN=IWKDAY
S. 228    C     PRD IS THE REVISED NETWORK DURATION.  JBOR (AFTER SECOND CALL TO PROGFL)
S. 229    C     CONTAINS THE REVISED VALUE PLUS ONE.
S. 230          PRD=JBOR-1
S. 231          IF(NSEL-NOACT)962,963,962
S. 232    C     PTR IS THE PHASE TIME RECOVERABLE.
S. 233      962 PTR=ITIMRC*(ORGDUR/(JOBDUR-1.))
S. 234          CONST=PTR*(1.-(TN/(JOBDUR-1.)))
S. 235          DO 94 J=1,NSEL
S. 236          I=JSORT(J)
S. 237          IF(KTIME(2,I)-IWKDAY)940,940,94
S. 238       94 CONTINUE
S. 239          WRITE(6,952) IWKDAY
S. 240          RETURN
S. 241      940 DO 950 J=1,NSEL
S. 242          I=JSORT(J)
S. 243          IF(KTIME(4,I)-IWKDAY)950,964,964
S. 244      950 CONTINUE
S. 245          WRITE(6,952) IWKDAY
S. 246      952 FORMAT(/,1X,'WORKDAY ',I4,' DOES NOT OCCUR DURING THE EXECUTION OF
               1 THIS SUBSET OF ACTIVITIES.  IT IS NOT VALID.')
S. 247          RETURN
S. 248      963 CONST=(1.-(TN/(JOBDUR-1.)))*ITIMRC
S. 249      964 IF(CONST)380,380,379
S. 250      380 WRITE(6,381) CONST
S. 251      381 FORMAT(/,1X,'DENOMINATOR OF WORK PROGRESS INDICATOR IS ',F6.2,' NO
               I COMPUTATIONS CAN BE MADE OR ACTION TAKEN.')
S. 252          DESTROY KTIME
S. 253          RETURN
S. 254      379 IF(CONST-(RPD-EPD))180,19,19
S. 255      180 IF(F)18,18,19
S. 256       18 MORE=CONST-(RPD-EPD)+.5
S. 257          WRITE(6,20) MORE
S. 258       20 FORMAT(//,1X,'THE REMAINING WORK WILL REQUIRE ',I3,' MORE DAYS THA
               IN CAN BE MADE UP.  REVISE YOUR SCHEDULE.')
S. 259          DESTROY KTIME
S. 260          RETURN
S. 261       19 WP=((CONST-(PRD-ORGDUR))/CONST)+.005
S. 262          IF(F)950,950,899
```

```
S. 263   C      DEPENDING UPON WHICH INDEX WAS REQUESTED, OUTPUT THE VALUE.
S. 264        850 GO TO (801,840,870),KKK
S. 265        801 CI=CI+.005
S. 266            WRITE(6,802) CI,JWKDAY
S. 267        802 FORMAT(30X,'**** THE VALUE OF THE COST COMPONENT IS 'F5.2,' FOR DA
                 1Y ',I4,' ****')
S. 268            DESTROY KTIME
S. 269            RETURN
S. 270        840 WP=WP+.005
S. 271            WRITE(6,841) WP,IWKDAY
S. 272        841 FORMAT(25X,'**** THE VALUE OF THE WORK PROGRESS COMPONENT IS ',F5.
                 12,' FOR DAY ',I4,' ****')
S. 273            DESTROY KTIME
S. 274            RETURN
S. 275        870 SI=WP*CI+.005
S. 276            WRITE(6,101)SI,JWKDAY
S. 277        101 FORMAT(/,31X,'**** THE VALUE OF THE STATUS INDEX IS ',F5.2,' FOR D
                 IAY ',I4,' ****')
S. 278            IF(ABS(CI-1.)-.1)205,205,302
S. 279        302 WRITE(6,303) CI
S. 280        303 FORMAT(/,10X,'**** NOTE -- VALUE OF COST COMPONENT IS ',F6.3,' ***
                 1*')
S. 281        205 IF(ABS(WP-1.)-.1)899,899,202
S. 282        202 WRITE(6,203) WP
S. 283        203 FORMAT(/,10X,'**** NOTE -- VALUE OF WORK PROGRESS COMPONENT IS ',F
                 16.3,' ****')
S. 284        899 DESTROY KTIME
S. 285            RETURN
S. 286            END
```

ICETRAN LISTING

```
S.   1          SUBROUTINE PRDGMD
S.   2          CDMMDN MDDE,NUMACT,NDACT,NUMAR,ICODE,IFLDW,DATEM,DATED,DATEY,DNUM
S.   3          CDMMDN NDHDL,JDBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   4          CDMMDN NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,DBXXX(5D)
S.   5          CDMMDN TABLE,HDLD,IALPHA(16),MDNTH(12),RLXXX(20),INTXXX(20)
S.   6          CDMMDN NARRDW(P),INCDDE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.   7          CDMMDN SDRT(P),JSDRT(P),KHDLD(P),MDUM(P),MARRDW(P),ICAL(P),IHDL(P)
S.   8          DYNAMIC ARRAY  NARRDW,INCDDE,INGRID,KTIME,IDTIME,IDNET,SDRT(D),
                JSDRT,KHDLD,MDUM,MARRDW,ICAL,IHDL
S.   9          DDUBLE PRECISIDN NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  10          DDUBLE PRECISION IALPHA,FMDNTH,DBXXX
S.  11          INTEGER F
S.  12          REAL RLXXX
S.  13          EQUIVALENCE (RLXXX(17),WP),(RLXXX(19),CI)
S.  14   P      EQUIVALENCE (DBXXX(12),STATRY(1))
S.  15          EQUIVALENCE (IPROG,INTXXX(10))
S.  16          DYNAMIC ARRAY STATRY
S.  17          KIND=INTXXX(2D)
S.  18          DESTRDY STATRY
S.  19          DEFINE STATRY,1D,FULL,LDW,STEP=5
S.  20          RELEASE STATRY
S.  21   C      TAKE THE LARGER DF THE TWD DATES AS THE TERMINAL DATE.
S.  22          IF(D-E)3D,40,50
S.  23       4D WRITE(6,41)
S.  24       41 FDRMAT(/,1X,'IDIDT -- BDTH DATES ARE THE SAME.')
S.  25          ERROR RETURN
S.  26       5D ILAS=D
S.  27          ISTA=E
S.  28          GD TO 21
S.  29       3D ILAS=E
S.  3D          ISTA=D
S.  31   C      CHECK TO SEE THAT TERMINAL DATE DDESN'T EXCEED JOBDUR.
S.  32       21 IF(ILAS-JDBDUR)1,2,2
S.  33        2 WRITE(6,3) ILAS
S.  34        3 FDRMAT(/,1X,'TERMINAL DATE FDR INDEX REQUEST, DAY ',I5,', IS BEYON
                D THE END OF THE PRDJECT.  REVISE REQUEST DATA.')
S.  35          ERRDR RETURN
S.  36        1 JI=ISTA
S.  37          I3=ILAS
S.  38          I4=0
S.  39          I5=0
S.  40          IF(ILAS-IPROG)62,62,61
S.  41       61 I2=IPROG
S.  42          GD TO 721
S.  43       62 I2=I3
S.  44      721 IF((ILAS-ISTA)-1DD)4,4,5
S.  45        4 INC=5
S.  46          WRITE(6,7) INC
S.  47        7 FORMAT(/,1X,'INDEX DATA WILL BE PRESENTED IN ',I2,' DAY INCREMENTS
```

```
                1.')
S.   48           JJ=1
S.   49           GO TO 8
S.   50         5 INC=10
S.   51           WRITE(6,7) INC
S.   52           JJ=1
S.   53         8 IF(ILAS-ISTA)20,10,10
S.   54        10 D=ISTA
S.   55           CALL PROGIX
S.   56   C       STORE THE STATUS INDEX AND ITS COMPONENTS IN THE ARRAY STATRY.
S.   57           IF(KIND-2)80,85,90
S.   58        80 STATRY(JJ)=CI
S.   59           GO TO 95
S.   60        85 STATRY(JJ)=WP
S.   61           GO TO 95
S.   62        90 STATRY(JJ)=WP*CI
S.   63        95 ISTA=ISTA+INC
S.   64           JJ=JJ+I
S.   65           GO TO 8
S.   66        20 CONTINUE
S.   67           IF(F)70,70,71
S.   68        71 DEFINE NARROW,JJ,FULL,LOW
S.   69           LINE(6)=JJ-1
S.   70           JTM=LINE(6)
S.   71   C       VALUES BY 10000. TO ELIMINATE DECIMAL FRACTIONS.
S.   72   C       PUT VALUES FROM STATRY INTO NARROW PRIOR TO CALLING PRGRAF -- MULTIPLY ALL
S.   73           DO 900 I=1,JTM
S.   74       900 NARROW(I)=STATRY(I)*10000.
S.   75           DESTROY STATRY
S.   76           WRITE(6,1003) NAME
S.   77      1003 FORMAT(///,46X,'**** INDEX PLOT FOR PROJECT ',A8,' ****',//)
S.   78           LINE(1)=J1
S.   79           LINE(2)=I2
S.   80           LINE(3)=I3
S.   81           LINE(4)=I4
S.   82           LINE(5)=I5
S.   83           ADD TO STACK(I,'PRGRAF')
S.   84           TRANSFER TO STACK
S.   85        70 RETURN
S.   86           END
```

ICETRAN LISTING

```
S.   1          SUBROUTINE PRSAVE
S.   2    C     THIS SUBROUTINE SAVES THE VALUE SET IN K5 AFTER A REQUEST FOR A STATUS
S.   3    C     INDEX HAS BEEN MADE.
S.   4          COMMON MODE,NUMACT,NOACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,ONUM
S.   5          COMMON NOHOL,JOBOUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   6          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,OBXXX(50)
S.   7          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.   8          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IDNET(P)
S.   9          COMMON SORT(P),JSORT(P),KHOLD(P),MOUM(P),MARROW(P),ICAL(P),IHOL(P)
S.  10          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IDNET,SORT(0),
               1JSORT,KHOLD,MOUM,MARROW,ICAL,IHOL
S.  11          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  12          DOUBLE PRECISION IALPHA,FMONTH,OBXXX
S.  13          INTXXX(20)=K5
S.  14          RETURN
S.  15          END
```

ICETRAN LISTING

```
S.   1          SUBROUTINE PROGET(IA,IB,IC)
S.   2    C     THIS SUBROUTINE RETRIEVES THE PROGRESS DATA WHICH HAS BEEN INPUT AND
S.   3    C     STORED IN NAME8.
S.   4    C     SETTING IA = 0 WILL CAUSE JUST THE IDPROG'S TO BE RETRIEVED.  IF IA = 1,
S.   5    C     THEN COLUMNS IB THRU IC OF NAME8 WILL BE RETRIEVED AND STORED IN KPROG.
S.   6          COMMON MODE,NUMACT,NDACT,NJMAR,ICODE,IFLOW,DATEM,DATED,DATEY,ONUM
S.   7          COMMON NDHOL,JDBOUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.   8          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,OBXXX(50)
S.   9          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.  10          COMMON NARROW(P),INCODE(P),INGRID(P),KTIME(P),IDTIME(P),IONET(P)
S.  11          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.  12          DYNAMIC ARRAY  NARROW,INCODE,INGRID,KTIME,IDTIME,IONET,SORT(0),
               1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.  13          DYNAMIC ARRAY IDPROG, KPROG
S.  14          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.  15          DOUBLE PRECISION IALPHA,FMONTH,OBXXX
S.  16          DOUBLE PRECISION NAME8
S.  17          EQUIVALENCE (OBXXX(9),NAME8)
S.  18    P     EQUIVALENCE (OBXXX(22),KPROG(1))
S.  19    P     EQUIVALENCE (OBXXX(23),IDPROG(1))
S.  20          NAME8=NAME
S.  21          CALL ADONUM(NAME8,8)
S.  22          DISK FILE INFO(2,NAME8,IEXIST,IFIRST,LAST)
S.  23          IF(IEXIST-1)20,20,51
S.  24    C     KPROG ARRAY DOESN'T EXIST.
S.  25       20 RETURN
S.  26       51 DISK GET(2,IFIRST,LINE(1),1,4)
S.  27          DEFINE IDPROG,NDACT,FULL,LOW
S.  28          DISK GET(2,LINE(1),IDPROG(1),0,0)
S.  29          IF(IA)53,53,54
S.  30    C     IF JUST THE IDPROG'S WERE REQUESTED, RETURN NOW.
S.  31       53 RETURN
S.  32    C     IF OTHER INFORMATION WAS REQUESTED RETRIEVE IT FROM NAME8 AND STORE IT IN
S.  33    C     THE ARRAY CALLED KPROG.
S.  34       54 CONTINUE
S.  35          DEFINE KPROG,9,POINTER,LOW
S.  36          DO 62 I=IB,IC
S.  37       62 DEFINE KPROG(I),NDACT,FULL,LOW
S.  38          NBYTE1=(IB-1)*4+1
S.  39          NBYTE2=IC*4
S.  40          DO 65 I=1,NDACT
S.  41          IF(IDPROG(I))65,65,63
S.  42       63 DISK GET(2,IDPROG(I),LINE(IB),NBYTE1,NBYTE2)
S.  43          DO 64 J=IB,IC
S.  44       64 KPROG(J,I)=LINE(J)
S.  45       65 CONTINUE
S.  46          RELEASE KPROG
S.  47          RELEASE IDPROG
S.  48          RETURN
```

ICETRAN LISTING

```
S.    1          SUBROUTINE PROGOL
S.    2   C                AS OF JAN. 26 1966
S.    3   C      THIS SUBROUTINE REVISED TO TREAT LAGS
S.    4   C      W.H. LINDER FOR SUBSYSTEM 'PROJECT' OF ICES, JULY 1966
S.    5   C      SHORTENED SUBROUTINE, DEBUGGED OCTOBER 1966
S.    6   C      FFLOW MAKES THE FORWARD SCHEDULING COMPUTATIONS
S.    7          COMMON MODE,NUMACT,NDACT,NUMAR,ICODE,IFLOW,DATEM,DATED,DATEY,ONUM
S.    8          COMMON NOHOL,JOBDUR,A,B,C,D,E,F,G,H,K1,K2,K3,K4,K5,K6,LINE(20)
S.    9          COMMON NAME,NAME1,NAME2,NAME3,NAME4,NAME5,A3(12),MAME,OBXXX(50)
S.   10          COMMON TABLE,HOLD,IALPHA(16),MONTH(12),RLXXX(20),INTXXX(20)
S.   11          COMMON NARROW(P),INCODE(P),INGRIO(P),KTIME(P),IDTIME(P),IDNET(P)
S.   12          COMMON SORT(P),JSORT(P),KHOLD(P),MDUM(P),MARROW(P),ICAL(P),IHOL(P)
S.   13          DYNAMIC ARRAY  NARROW,INCODE,INGRIO,KTIME,IDTIME,IDNET,SORT(D),
                 1JSORT,KHOLD,MDUM,MARROW,ICAL,IHOL
S.   14          DOUBLE PRECISION NAME,NAME1,NAME2,NAME3,NAME4,NAME5,BLANK,MAME
S.   15          ODUBLE PRECISION IALPHA,FMONTH,OBXXX
S.   16   C      INITIALIZE STARTS AS 1 AND FINISHES AS 1 + DURATION
S.   17   C      BRANCH ON DATEM FOR WORK DAY START
S.   18          IF(DATEM) 2,2,3
S.   19   2      NOAY=DATED
S.   20          GO TO 4
S.   21   3      NOAY=1
S.   22        4 DO 10 I=1,NDACT
S.   23          IF(KTIME(2,I))7,6,7
S.   24        6 KTIME(2,I)=NOAY
S.   25        7 IF(KTIME(4,I))10,8,10
S.   26        8 KTIME(4,I)=KTIME(1,I)+KTIME(2,I)
S.   27       10 CONTINUE
S.   28   C      SEE IF THIS IS A NETWORK WITH NO ARROWS. IF SO, SKIP THESE CALCULATIONS
S.   29          IF(NUMAR)51,51,52
S.   30   C      NOW WORK DOWN ARROW LIST CALCULATING ES AND EF FROM EF OF SOURCES
S.   31       52 DO 50 I=1,NUMAR
S.   32          K=NARROW(1,I)
S.   33          KK=NARROW(2,I)
S.   34          IF(MODE)100,99,100
S.   35   99     LAG=NARROW(3,I)
S.   36          GO TO 98
S.   37   100    LAG=0
S.   38   C      SEE IF ES AT ARROW HEAD IS LESS THAN EF OF ARROW TAIL ACTIVITY
S.   39   98     IF(KTIME(4,K)-KTIME(2,KK)+LAG)50,50,80
S.   40   C      YES, SO PUT EF OF ARROWTAIL ACTIVITY INTO ES OF ARROWHEAD ACTIVITY
S.   41   80     KTIME(2,KK)=KTIME(4,K)+LAG
S.   42   C      NOW ADD DURATION TO GET EF, IF EF HAS NOT BEEN SET LARGER
S.   43          IF(KTIME(4,KK)-KTIME(2,KK)-KTIME(1,KK))82,50,50
S.   44       82 KTIME(4,KK)=KTIME(2,KK)+KTIME(1,KK)
S.   45   50     CONTINUE
S.   46       51 RELEASE NARROW
S.   47          RELEASE KTIME(1)
S.   48          RELEASE KTIME(2)
```

```
S.  49    C      NOW FIND JOB DURATION THE LATEST(GREATEST) EARLY FINISH
S.  50           JOBDUR=1
S.  51           DO 60 I=1,NOACT
S.  52           IF(JOBDUR-KTIME(4,I))65,60,60
S.  53    65     JOBDUR=KTIME(4,I)
S.  54    60     CONTINUE
S.  55           RELEASE KTIME(4)
S.  56           RETURN
S.  57           END
```

```
S.  49    C      NOW FIND JOB DURATION THE LATEST(GREATEST) EARLY FINISH
S.  50           JOBDUR=1
S.  51           DO 60 I=1,NOACT
S.  52           IF(JOBDUR-KTIME(4,I))65,60,60
S.  53    65     JOBDUR=KTIME(4,I)
S.  54    60     CONTINUE
S.  55           RELEASE KTIME(4)
S.  56           RETURN
S.  57           END
```

```
/ASSIGN  2=PROJECT
/ASSIGN  3=ICES
/RESTORE SYS1

 CDP

 SYSTEM 'PROJ' 'LINDER'

 REPLACE 'SELINX'  $  SELINX CALLS COMPUTATIONAL PROGRAMS FOR STATUS INDEX.

 CONDITION REAL 'E' GE 1  $ ENTRY HERE IMPLIES WE HAVE MULTIPLE DATES.

       EXECUTE 'PROGMD'  $ CALL THE MULTIPLE DATE SUBROUTINE.

       EXECUTE 'PROGDJ'  $ DESTROY THE JSORT ARRAY.

 OTHERWISE  $ ENTRY HERE IMPLIES WE HAVE A SINGLE DATE.

       EXECUTE 'PROGIX'  $ EXECUTE COMPUTATIONAL SUBROUTINE TO FIND INDEX.

       EXECUTE 'PROGDJ'  $ DESTROY THE JSORT ARRAY.

 END CONDITION

 RETURN

 FILE
1

 FINISH


 GOOD-BYE


 ELAPSED TIME=  00.00 MINUTES.
```

```
/ASSIGN  2=PROJECT
/ASSIGN  3=ICES
/RESTORE SYS1

 COP

 SYSTEM 'PROJ' 'LINDER'

 REPLACE 'PROGRESS'

 IGNORE 'REP' 'DA' 'AS' 'OF' 'FOR' 'W'

 PRESET REAL 'H' EQ -1  $ SET SO AS TO SKIP PROLEC IN 'IPDATE'.

 EXISTENCE 'PRO' 'FIN' 'STA' 'COS' SET 'I2' STA 5

 CONDITION 'I2' LE 4
       DATA CHECK SET 'K4'  $ SEE IF THERE'S A DATE GIVEN
       CONDITION INTEGER 'K4' EQ -1
             MESSAGE ' DATE MISSING IN *REPORT* COMMAND.  NO ACTION TAKEN.'
             RETURN  $ CEASE PROCESSING THIS COMMAND.
       OTHERWISE
       END CONDITION
       CALL 'IPDATE'
       CONDITION REAL 'B' NE 0.
             EXECUTE 'PRPORK'
       OTHERWISE
             MOVE 'A' TO 'D'
       END CONDITION
       PRESET 'K1' EQ 1
       EXECUTE 'PROGST'  $ INITIALIZE FOR PROGRESS INPUT - CREATE NAMES.
       REPEAT TABULAR
       IGNORE 'ACT' 'DAY'
       DATA CHECK SET 'K4'  $ IS THIS THE LAST DATA CARD.
             CONDITION INTEGER 'K4' LT 2  $ NOT SATISFIED IF NEXT DATA IS ACT. N
                   IGNORE 'LAS'
                   REPEAT EXIT
             END CONDITION OPTIONAL
       NO ID REAL 'A' REQ  $ FIRST NODE NUMBER.
```

```
PRESET REAL 'B' EQ 0.  $ SET B=0. SO YOU CAN CHECK IT LATER.
      CONDITION 'MODE' EQ 1  $ IS NETWORK A/A
            NO ID REAL 'B' REQ  $ SECOND NODE NUMBER.
      END CONDITION OPTIONAL
PRESET 'K1' EQ 2  $ CHECK AND STORE ACTIVITY NUMBER
EXECUTE 'PROGST'
      CONDITION 'K3' LT 1  $ SKIP DATA CARD IF BAD ACTIVITY NO.
      OTHERWISE  $ IF ACTIVITY NO. IS VALID, CONTINUE.
            CONDITION 'I2' EQ 1  $ GENERAL PURPOSE INPUT COMMAND.
                  CALL 'DATAST'  $ SUBROUTINE CDL TO PROCESS PROGRESS DAT
            OR CONDITION 'I2' EQ 2  $ COMMAND IS REPORT FINISH
                  IGNORE 'F'
                  CALL 'IPDATE'
                        CONDITION REAL 'B' EQ 0.
                              MOVE 'A' TO 'D'
                        OTHERWISE
                              EXECUTE 'PRPORK'
                        END CONDITION
                  PRESET 'K1' EQ 3
                  EXECUTE 'PROGST'  $ STORE FINISH DATE IN LINE2.
            OR CONDITION 'I2' EQ 3  $ COMMAND IS REPORT START
                  IGNORE 'S'
                  CALL 'IPDATE'
                        CONDITION REAL 'B' EQ 0.
                              MOVE 'A' TO 'D'
                        OTHERWISE
                              EXECUTE 'PRPORK'
                        END CONDITION
                  PRESET 'K1' EQ 4  $ STORE START DATE IN LINE3.
                  EXECUTE 'PROGST'
            OR CONDITION 'I2' EQ 4  $ COMMAND IS REPORT COST
                  IGNORE 'C'
                  NO ID REAL 'C'
```

```
                    PRESET 'K1' EQ 5  $ STORE COST DATA.
                    EXECUTE 'PROGST'
                END CONDITION OPTIONAL
                PRESET 'K1' EQ 6
                EXECUTE 'PROGST'  $ STORE INFO ON THE DISK
            END CONDITION
        END REPEAT TABULAR
    PRESET 'K1' EQ 7  $ STORE ID OF UPDATED ACTIVITY IN DIRECTORY.
    EXECUTE 'PROGST'
    OTHERWISE  $ IF INVALID COMMAND PRINT ERROR MESSAGE.
        MESSAGE ' INVALID OBJECT NAME FOR **REPORT** COMMAND.  NO ACTION TAKEN.'
    END CONDITION
    NEXT RECORD
    NEW COMMAND
    RETURN
    FILE
1
    FINISH

    GOOD-BYE

    ELAPSED TIME=  00.00 MINUTES.
```

```
/ASSIGN  2=PROJECT
/ASSIGN  3=ICES
/RESTORE SYS1

 COP

 SYSTEM 'PROJ' 'LINDER'

 REPLACE 'DATAST'                        $ SUBROUTINE COL TO PROCESS PROGRESS DAT

                         REPEAT

 $  SEE IF THE END OF CARD FOLLOWS.  IF YES, EXIT THE REPEAT LOOP.

                         DATA CHECK SET 'K6'

                         CONDITION INTEGER 'K6' EQ -1

                               REPEAT EXIT

                         OTHERWISE

                         END CONDITION

                         MODIFIER 'FIN'

                               CALL 'IPDATE'

                               CONDITION REAL 'B' EQ 0.

                                     MOVE 'A' TO 'D'

                               OTHERWISE

                                     EXECUTE 'PRPORK'

                               END CONDITION

                               PRESET 'K1' EQ 3

                               EXECUTE 'PROGST'  $ STORE FINISH DATE.

                         OR MODIFIER 'STA'

                               CALL 'IPDATE'

                               CONDITION REAL 'B' EQ 0.

                                     MOVE 'A' TO 'D'

                               OTHERWISE

                                     EXECUTE 'PRPORK'

                               END CONDITION

                               PRESET 'K1' EQ 4  $ STORE START DATE.

                               EXECUTE 'PROGST'

                         OR MOD 'COS'
```

```
                NO ID REAL 'C'
                PRESET 'K1' EQ 5  $ STORE COST DATA.
                EXECUTE 'PROGST'
          OR CONDITION 'K4' EQ 29
                NO ID REAL 'C'
                PRESET 'K1' EQ 5  $ STORE COST DATA.
                EXECUTE 'PROGST'
          OR CONDITION 'K4' EQ 31
                CALL 'IPDATE'
                CONDITION REAL 'B' EQ 0.
                      MOVE 'A' TO 'D'
                OTHERWISE
                      EXECUTE 'PRPORK'
                END CONDITION
                PRESET 'K1' EQ 4  $ STORE START DATA.
                EXECUTE 'PROGST'
          OR CONDITION 'K4' EQ 32
                CALL 'IPDATE'
                CONDITION REAL 'B' EQ 0.
                      MOVE 'A' TO 'D'
                OTHERWISE
                      EXECUTE 'PRPORK'
                END CONDITION
                PRESET 'K1' EQ 3  $ STORE FINISH DATE.
                EXECUTE 'PROGST'
          OTHERWISE  $ NO MODIFIER INPUT.
$  THIS PORTION OF THE CDL IS USED WHEN THERE ARE NO MODIFIERS GIVEN WITH DATA.
          MESSAGE 'NO MODIFIERS.  ASSUMED INPUT ORDER TO BE START
, FINISH, COST (OPTIONAL).'
          CALL 'IPDATE'
          CONDITION REAL 'B' EQ 0
                MOVE 'A' TO 'D'
          OTHERWISE
```

```
                        END CONDITION
                        PRESET 'K1' EQ 4
                        EXECUTE 'PROGST'  $ STORE THE START DATA.
                        CALL 'IPDATE'
                        CONDITION REAL 'B' EQ 0
                              MOVE 'A' TO 'D'
                        OTHERWISE
                              EXECUTE 'PRPORK'
                        END CONDITION
                        PRESET 'K1' EQ 3
                        EXECUTE 'PROGST'  $ STORE FINISH DATA.
                        DATA CHECK SET 'G'
                        CONDITION INTEGER 'G' GE 2
$  G LARGER THAN 1 IMPLIES THAT A NUMBER -- HERE A COST -- FOLLOWS.
                              NO ID REAL 'C' REQ
                              PRESET 'K1' EQ 5
                              EXECUTE 'PROGST'  $ STORE COST DATA IF IT EXISTS.
                        OR CONDITION INTEGER 'G' EQ -1
                        OTHERWISE
MESSAGE ' *** ERROR IN PREVIOUS DATA CARD.  PROCESSING CONTINUES ***'
                        END CONDITION
                        REPEAT EXIT
                        END CONDITION
                        END MODIFIER BLOCK
                        END REPEAT
RETURN
FILE
1
FINISH


GOOD-BYE


ELAPSED TIME=  00.00 MINUTES.
```

APPENDIX B

There is a special convention for writing commands in a text such as this. Words in parenthesis are optional and may be omitted. The set of characters "('projname')" is, of course, the name of the particular project in question enclosed in quotes. This information may be omitted if the project name is available from a previous command. Finally, any word appearing in small letters (such as "date") indicates where alphameric or numeric data must be provided.

BIBLIOGRAPHY

1. Baumgartner, John S., Project Management, Richard D. Irwin, Homewood, Illinois, 1963

2. Genest, Bernard-Andre (Editor), The Use of ICES PROJECT, Civil Engineering Department, Massachusetts Institute of Technology, Cambridge, 1967

3. Hackney, John W., Control and Management of Capital Projects, John Wiley & Sons, Inc., New York, 1965